# ระบบตรวจสอบและวิเคราะห์ความปลอดภัยในระบบเครือข่ายท้องถิ่น

# Local Computer Security Monitoring and Analysis


โดย

นาย จักรพงษ์ เตชะจงเจริญ

นาย ทวีทรัพย์ อภิวัฒนาพงศ์




ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขา วิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

# ระบบตรวจสอบและเฝ้าดูแพ็กเกจในระบบเครือข่ายท้องถิ่น

# Local Computer Network Monitoring and Analysis


โดย

นาย จักรพงษ์ เตชะจงเจริญ 37014049

นาย ทวีทรัพย์ อภิวัฒนาพงศ์ 37014139

อาจารย์ที่ปรึกษา

อาจารย์ ธนา หงษ์สุวรรณ


ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขา วิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ ปีการศึกษา 2540

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง ระบบตรวจสอบและเฝ้าดูแพ็กเกจในระบบเครือข่ายท้องถิ่น

ผู้จัดทำ

1. นาย จักรพงษ์ เตชะจงเจริญ
2. นาย ทวีทรัพย์ อภิวัฒนาพงศ์

..... อาจารย์ที่ปรึกษา

( อาจารย์ ธนา หงษ์สุวรรณ )

# โครงงานระบบตรวจสอบและเฝ้าดูแพ็กเกจในระบบเครือข่ายท้องถิ่น

จักรพงษ์ เตชะจงเจริญ
ทวีทรัพย์ อภิวัฒนาพงศ์

อ. ธนา หงษ์สุวรรณ
ปีการศึกษา 2540

บทคัดย่อ

ในปัจจุบันนี้ การใช้งานคอมพิวเตอร์ (Computer) ในรูปแบบของเครือข่าย ( Network ) มีบทบาทอย่างมากในการเพิ่มประสิทธิภาพการทำงาน โดยเฉพาะองค์กรใหญ่ ๆ จะได้ประโยชน์จากการใช้ทรัพยากรร่วมกัน หรือ การทำงานเป็นระบบเครือข่าย ดังนั้นจึงได้ทำการพัฒนาโปรแกรม (Program) ขึ้นเพื่อใช้เฝ้าดู และวิเคราะห์แพ็กเกจ (Packet) ในระบบเครือข่ายท้องถิ่น (LAN : Local Area Network) โดยตรวจสอบแพ็กเกจที่ได้รับจากสื่อระบบ ผ่านการทำงานของแพ็กเกจไดรเวอร์ (Packet Driver) ซึ่งจะรับแพ็กเกจจากการ์ดเชื่อมต่อเน็ตเวิร์ค (NIC : Network Interface Card) เข้ามาวิเคราะห์ถึงส่วนต่าง ๆ ที่จำเป็นต่อการศึกษาการทำงาน และตรวจสอบระบบเครือข่าย โดยวิเคราะห์ในด้านต่าง ๆ เช่น ปริมาณการใช้งาน (Utilization) , การกระจายโปรโตคอล (Protocol Distribution) , การกระจายของขนาดเฟรม (Frame Size Distribution) , การเก็บข้อมูลเป็นช่วงเวลา รวมถึง การเฝ้าดูการส่งและรับแพ็กเกจของเครื่องใด เครื่องหนึ่งในระบบเครือข่าย

โครงงานนี้ได้พัฒนามาจากโครงงานของปีที่แล้ว โดยได้เพิ่มเติมส่วนต่าง ๆ จากเดิม และ แก้ไขการทำงานบางส่วนที่ไม่สมบูรณ์ โดยได้ปรับปรุงส่วนติดต่อกับผู้ใช้ ส่วนแสดงผลกราฟ และ เพิ่มหน้าที่ต่าง ๆ เพื่อความสะดวกในการวิเคราะห์เครือข่าย

# LOCAL COMPUTER NETWORK MONITORING AND ANALYSIS

Jakaphong Tejachongcharoen

Taweesup Apiwattanapong

Mr. Thana Hongsuwan

1997

Abstract

Nowadays, many large organizations use the computers in networking system for sharing the resources and increasing efficiency. In order to do this, the program that monitors and analyses local area network is developed. The program calls Packet driver which handles the activities of NIC (Network Interface Card). And the NIC captures packets from network. Then the program analyses captured packet and shows the statistic.

This project was developed from last year project by including new functions, modifying some existing incomplete functions ,and improving user interface.

# สารบัญ

หน้า

หน้า

หน้า

# สารบัญภาพ

หน้า

# บทที่ 1
# บทนำ

## 1.1 ความสำคัญและที่มา

ในการใช้งานคอมพิวเตอร์ในสภาพแวดล้อมที่เป็นเครือข่ายนั้นแตกต่างจากการใช้งานในสภาพแวดล้อมที่มีผู้ใช้เพียงคนเดียว ในสายส่งข้อมูลนั้นจะมีแพ็กเกจไหลอยู่ และมีโปรโตคอลสื่อสาร เพื่อใช้ส่งข้อมูล และ การจัดสรรทรัพยากรในระบบ เช่น เน็ตแวร์ แลน (Netware LAN) เมื่อมีการใช้งานเครือข่ายเพิ่มขึ้น เราจึงต้องการรายละเอียดข้อมูลเกี่ยวกับประสิทธิภาพการใช้งานของโปรโตคอล (Protocol) ปัญหาการใช้งาน การทดสอบ การทำให้ได้ผลดีที่สุด และเพื่อใช้ประกอบในการเรียนรู้ถึงกระบวนการต่าง ๆ ที่เกิดขึ้นภายในระบบ โดยสามารถเข้าใจถึงการทำงานจากการเฝ้าดูแพ็กเกจที่วิ่งในเครือข่าย สามารถนำไปใช้แก้ปัญหาที่เกิดขึ้นภายในระบบเครือข่าย ซึ่งจะต้องตรวจสอบถึงสาเหตุต่าง ๆ ที่เกิดขึ้น โดยดูจากข้อมูลต่าง ๆ ที่ได้ในเครือข่าย เช่น ประสิทธิภาพการใช้งานเครือข่าย การสนทนาระหว่างเครื่อง ปริมาณการใช้งานของแต่ละโปรโคคอล และ การกระจายของขนาดเฟรม ซึ่งข้อมูลต่าง ๆ นี้นำมาใช้ประกอบในการแก้ไขปัญหาของระบบได้เป็นอย่างดี ทั้งนี้โปรแกรมและ เครื่องมือต่างๆ จำต้องสั่งซื้อมาจากต่างประเทศซึ่งมีราคาแพง และอาจไม่ตรงกับความต้องการมากนัก จึงคิดว่าน่าจะสามารถพัฒนาโปรแกรมนี้เองได้

## 1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อพัฒนาซอฟต์แวร์ (Software) ที่สามารถตรวจสอบและเฝ้าดู ระบบเครือข่ายเพื่อการควบคุม และดูแลระบบโดยเจ้าหน้าที่ดูแลระบบ

1.2.2 เพื่อการศึกษา การใช้งานของโปรโตคอล ในแต่ละเลเยอร์ เพื่อให้เข้าใจการทำงานของโปรโตคอล

1.2.3 เพื่อจัดทำการเก็บสถิติเพื่อดูระบบเครือข่ายถึงการใช้งานของระบบ

1.2.4 เพื่อประกอบในการปรับปรุงระบบเครือข่ายให้ทราบถึงจุดบกพร่องของระบบ

1.2.5 เพื่อช่วยในการพัฒนาระบบ ให้สามารถสื่อสารข้อมูล ได้รวดเร็วเพิ่มขึ้น โดยการหาจุดบกพร่องแล้วนำมาพัฒนาระบบ

## 1.3 ขอบเขตของโครงงาน

1.3.1 เป็นซอฟต์แวร์คอมพิวเตอร์ที่ทำงานบนระบบเครือข่ายท้องถิ่น (LAN)

1.3.2 สามารถเก็บแพ็กเกจเพื่อนำมาวิเคราะห์ ตรวจสอบเฝ้าดู โดยวิเคราะห์โปรโตคอลหลัก ๆ ในแต่ละเลเยอร์ ดังนี้

1.3.2.1 ดาต้าลิงค์เลเยอร์ (Datalink Layer)

- ไออีอีอี 802.2 (IEEE 802.2)
- ไออีอีอี 802.3 (IEEE 802.3)

- อีเทอร์เนตทู (Ethernet II)
- อีเทอร์เนตสแนบ (Ethernet SNAP)

1.3.2.2 เน็ตเวิร์คเลเยอร์ (Network Layer)

- ไอพี (IP)
- ไอพีเอ็กซ์ (IPX)
- เออาร์พี (ARP)

1.3.2.3 ทรานสปอร์ตเลเยอร์ (Transport Layer)

- ทีซีพี (TCP)
- ยูดีพี (UDP)
- ไอซีเอ็มพี (ICMP)
- เอสพีเอ็กซ์ (SPX)

1.3.2.4 อัพเปอร์เลเยอร์ (Upper Layer)

- เทลเนต (TELNET)
- เอฟทีพี (FTP) เอฟทีพีดาต้า (FTP_DATA)
- อาร์ล็อกอิน (RLOGIN)
- อื่น ๆ

1.3.3 สามารถกรองแพ็กเกจได้โดยกรองในรูปแบบ ลักษณะของแพ็กเกจ และค่าของฟิลด์ (Field) ในแพ็กเกจนั้น รวมถึงการกรองโปรโตคอลก่อนที่จะแสดงผลบนหน้าจอ

1.3.4 แสดงภาพภายในเน็ตเวิร์คโดยกล่าวถึงระดับการใช้งานของเครือข่าย (Utilize) อัตราการเกิดข้อผิดพลาด (Error Rate) จำนวนแพ็กเกจต่อวินาที (Packet per second) กิโลไบต์ต่อวินาที (Kilobyte per second) โปรโตคอลที่ใช้อยู่ (Protocol inuse) การกระจายของขนาดแพ็กเกจ (Packet size Distribution) การกระจายการใช้งานของโปรโตคอล (Protocol Distribution)

1.3.5 แสดงการสนทนาระหว่างโฮส (Host Conversion) โดยแสดงการใช้งานของแต่ละโฮสโดยเปรียบเทียบทั้งแบบโฮสเดี่ยว และระหว่างโฮสด้วยกัน

1.3.6 สามารถตรวจสอบโฮสว่ามีการตอบสนองการใช้งานหรือไม่ และตรวจสอบระยะเวลาที่ตอบรับเพื่อตรวจสอบดูความหนาแน่นของการใช้งานเครือข่ายของระบบแต่ละโฮส

1.3.7 สามารถเก็บข้อมูลเพื่อเป็นสถิติของปริมาณการใช้งานของเครือข่ายของโดยรวม ขนาดของแพ็กเกจ โปรโตคอล และสามารถดูข้อมูลย้อนหลังได้

1.3.8 สามารถจำลองการทำงานของเครือข่ายโดยสร้างแพ็กเกจแล้วส่งไปในเครือข่าย

1.3.9 วิเคราะห์โปรโตคอล โดยถอดรหัสแพ็กเกจ และแสดงโปรโตคอล ที่ใช้ในแพ็กเกจในแต่ละเลเยอร์

1.3.10 สามารถกำหนดชื่อให้กับฮาร์ดแวร์แอดเดรสของเครื่องภายในเครือข่ายได้

1.3.11 สามารถเฝ้าดูการรับส่งข้อมูลจากเครือข่ายของเครื่องใดเครื่องหนึ่งโดยเฉพาะได้

**1.4 ประโยชน์ที่คาดว่าจะได้รับ**

1.4.1 อำนวยความสะดวกรวดเร็วในการติดต่อสื่อสารระหว่างกลุ่มผู้ใช้

1.4.2 ช่วยแบ่งเบาหน้าที่ผู้ดูแลระบบ

1.4.3 ประหยัดทรัพยากรบุคคลในการทำงาน

1.4.4 แก้ปัญหาทางด้านการติดต่อระหว่างบุคคล โดยใช้คอมพิวเตอร์ช่วยในการสื่อสาร

1.4.5 ศึกษาโครงสร้างของแพ็กเกจ และการสนทนาระหว่างโฮส

**1.5 ส่วนประกอบเนื้อหาทั้งหมดของโครงงาน**

ในปริญญานิพนธ์นี้ ประกอบด้วย 6 บทด้วยกันซึ่งครอบคลุม ความรู้และ ทฤษฎีพื้นฐาน การวางแผนการออกแบบ และการสร้าง การทดลอง และผลของการทดลอง และ บทวิจารณ์และสรุป

บทที่ 1 บทนำ จะกล่าวถึงวัตถุประสงค์ ความสำคัญ และ ที่มาของโครงงาน รวมถึงขอบเขตของโครงงาน และ ประโยชน์ที่ได้รับจากโครงงานนี้ รวมทั้งส่วนประกอบเนื้อหาของปริญญานิพนธ์นี้

บทที่ 2 ความรู้และทฤษฎีพื้นฐาน เกี่ยวกับระบบเครือข่าย คอมพิวเตอร์ เช่น โครงสร้างระบบโอเอสไอ (OSI) โครงสร้างของอีเทอร์เนต (Ethernet) โปรโตคอลในเลเยอร์ (Layer) ต่าง ๆ

บทที่ 3 ความรู้และทฤษฎีพื้นฐาน เกี่ยวกับ หลักการของเน็ตเวิร์คมอนิเตอริง การวัดประสิทธิภาพและปัญหาของเครือข่าย การใช้งานของแพ็กเกจไดรเวอร์ ซึ่งในส่วนของโปรโตคอลในเลเยอร์ จะกล่าวถึงรูปแบบส่วนหัวของแพ็คเกจ และ หลักการของโปรโตคอลในเลเยอร์ ทีซีพี/ไอพี จะกล่าวถึงรูปแบบโปรโตคอล ไอพี ไอซีเอ็มพี ทีซีพี ยูดีพี

บทที่ 4 การวางแผน การออกแบบ และ การพัฒนา มีเนื้อหาถึงหลักการเบื้องต้นในการออกแบบซอฟต์แวร์ การวางแผน และพัฒนาระบบ วิธีการทำงานของโปรแกรม ซึ่งจะกล่าวถึงรายละเอียดของแต่ละโมดูล (module) ที่ใช้ในการสร้างโปรแกรม รวมถึงโครงสร้างข้อมูลแต่ละโมดูล ในแต่ละโมดูลนั้นประกอบด้วยการเก็บสถิติในรูปแบบระยะสั้น และ ระยะยาว การใช้งานของแต่ละโฮส (host) การเตือนภัยเมื่อมีปัญหาในเน็ตเวิร์ค การกรองแพ็กเกจ การเก็บแพ็กเกจ การส่งแพ็กเกจบนเครือข่าย และสุดท้ายการถอดรหัสโปรโตคอล ตลอดถึงบางส่วนในปริญญานิพนธ์ที่ได้ให้ข้อมูลเกี่ยวกับตัวอย่างปัญหา การแก้ปัญหา และการเพิ่มประสิทธิภาพของเน็ตเวิร์ค

บทที่ 5 การทดลองและ ผลการทดลอง ผลการทำงานของโปรแกรมแต่ละส่วน ปัญหาและอุปสรรค ที่พบในระหว่างการทำงาน

บทที่ 6 บทวิจารณ์ และบทสรุป กล่าวถึงความคิดเห็นในการพัฒนาโปรแกรมต่อไป และสรุปโครงงาน

ภาคผนวก ก ประกอบด้วยคำศัพท์ที่ใช้ในปริญญานิพนธ์นี้ คำแปลและความหมาย ภาคผนวก ข รูปแบบส่วนหัวของแต่ละโปรโตคอล ภาคผนวก ค การกำหนดตัวเลขที่ใช้ในอีเทอร์เนต

**1.6 รูปแบบที่ใช้ในวิทยานิพนธ์**

ทุกตัวเลขในเอกสารนี้เขียนในรูปแบบเฉพาะ โดยเลขฐานสิบเขียนตรง ๆ เช่น 11 และเลขฐานสิบหก เขียนตัวนำในรูป "0x" เช่น 0x0B ทุก ๆ ค่าที่อ้างอิงกับเน็ตเวิร์คฮาร์ดแวร์แอดเดรส (ต้นทาง ปลายทาง) จะเขียนในรูปแบบเลขฐานสิบหกโดยไม่มีตัวนำ เช่น AA-BB-CC-DD-EE-FF

เมื่อมีคำศัพท์เทคนิคใหม่ ที่ไม่ปรากฏมาก่อนจะเขียนในรูปแบบคำอ่านพร้อมกับคำอธิบายในวงเล็บ ที่เป็นภาษาอังกฤษ ซึ่งจะพบในภาคผนวก ก โดยเก็บรวบรวมคำศัพท์เทคนิคไว้ คำศัพท์ที่ใช้เฉพาะจะพิมพ์ด้วยตัวหนา

# บทที่ 2

# สถาปัตยกรรมเครือข่าย

ความรู้ที่ต้องใช้ในการดำเนินโครงงานประกอบด้วย

## 2.1 สถาปัตยกรรมเครือข่าย ระดับชั้นโปรโตคอล และ โอเอสไอโมเดล

## (Network Architectures , Layer Protocol and OSI Model)

เป็นการยากที่จะให้อุปกรณ์ในระบบเครือข่ายสามารถใช้งานร่วมกันได้ ถ้าปราศจากมาตรฐานในการเข้ากันได้ (Compatible) ดังนั้นในปี 1984 อินเตอร์เนชั่นแนล์สแตนดาร์ดออร์กาไนซ์เซชั่น (ISO : International Standard for Organization) ได้กำหนดสถาปัตยกรรมในการเชื่อมต่อเข้าด้วยกัน ในระบบเครือข่ายคอมพิวเตอร์ เรียกว่าโอเพ่นซิสเต็มอินเตอร์คอนเนคชั่น (OSI : open system interconnection)

### 2.1.1 สถาปัตยกรรมเครือข่าย

ก่อนที่จะเริ่มกล่าวถึงสถาปัตยกรรมเครือข่ายและโปรโตคอล คำว่าสถาปัตยกรรมเครือข่ายนั้นมักจะถูกใช้เพื่ออธิบายลักษณะของเครือข่าย โดยพิจารณาจากฮาร์ดแวร์และซอฟท์แวร์ ดาต้าลิงค์คอนโทรล (datalink control) มาตรฐาน โทโพโลจี (topology) และ โปรโตคอล โปรโตคอล หมายถึง ระเบียบแบบแผนในการที่ส่วนประกอบของเครือข่ายใช้สร้างการสื่อสาร แลกเปลี่ยนข้อมูล และสิ้นสุดการสื่อสาร คำว่าโทโพโลจี ใช้เพื่อบอกถึง รูปแบบเครือข่ายว่ามีการเชื่อมต่ออย่างไร

### 2.1.2 ระดับชั้นโปรโตคอล

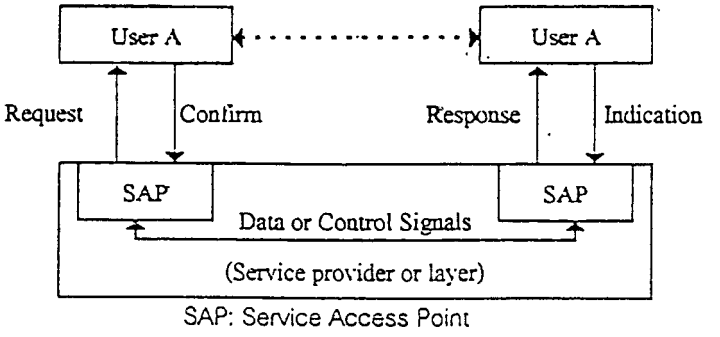
ในการออกแบบโปรโตคอลนั้นจะต้องมี

- แบ่งแยกระบบที่ซับซ้อนเป็นระบบเล็ก ๆ มีความเข้าใจในแต่ละส่วน
- เป็นมาตรฐาน ระหว่างหน้าที่ของเลเยอร์
- แต่ละเลเยอร์ในระดับเดียวกันจะต้องมีหน้าที่เหมือนกัน
- การเปลี่ยนอุปกรณ์ในเลเยอร์หนึ่งจะไม่มีผลในเลเยอร์อื่น ๆ

โดยแท้จริงแล้วระดับชั้นโปรโตคอล จะยึดถือแบบที่สามารถให้ระบบต่างกันสามารถสื่อสารกันได้อย่างง่ายและกว้างขวาง โดยปราศจากการเปลี่ยนแปลงในการสื่อสารแต่ละเลเยอร์ หรือเปลี่ยนแปลงให้น้อยที่สุด

*การสื่อสารระหว่างเลเยอร์*

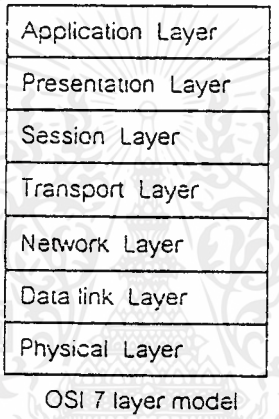
ในระดับชั้นโปรโตคอล และโอเอสไอโมเดล ระบบหนึ่ง ๆ ประกอบด้วยเลเยอร์ ตามรูป 2.1 แสดงถึง เลเยอร์ชั้นที่หนึ่งจัดการให้บริการระหว่าง User A และ User B การสื่อสารระหว่างผู้ใช้ผ่านการบริการไปถึง แอดเดรส หรือตัวกำหนด มักจะเรียกว่า เซอร์วิสแอคเซสพอยต์ (SAP : Service Access Point) ซึ่งจะจัดการบ่งบอกถึงอุปกรณ์นั้นให้เป็นหนึ่งเดียวไม่ซ้ำ




*รูปที่ 2.1 โอเอสไอเลเยอร์ และ การจัดเตรียมบริการ*

### 2.1.3 โอเอสไอโมเดล

เมื่อกำหนดหลักการของโปรโตคอลจึงได้แบ่งโปรโตคอลเป็นระดับชั้นต่าง ๆ ได้เป็น 7 ชั้น ดังนี้




*รูปที่ 2.2 การแบ่งเลเยอร์ในโอเอสไอโมเดล*

**1. ฟิสิคอล เลเยอร์** (Physical layer) ทำหน้าที่กำหนดการสื่อสารทางกายภาพ ระดับสัญญาณทางไฟฟ้า วัสดุตัวนำที่ใช้ในการสื่อสาร การต่อเชื่อมทางกายภาพ ตัวอย่างอุปกรณ์ในเครือข่ายที่ทำงานในชั้นนี้ ได้แก่ รีพีทเตอร์ (Repeater) หรือ ตัวทวนสัญญาณ

**2. ดาต้าลิงค์ เลเยอร์** ( Data link layer ) ทำหน้าที่ควบคุมการสื่อสาร แบบจุดต่อจุดที่ติดกัน (Point to Point) ให้สามารถรับส่งข้อมูลได้อย่างถูกต้อง ปราศจากข้อมูลที่ผิดพลาด ในโหนดที่ติดกัน มาตรฐานในระดับนี้ ได้แก่ ดีไอเอ็กซ์อีเทอร์เนต ไออีอีอี 802.5 เป็นต้น อุปกรณ์ในเครือข่ายที่ทำงานในชั้นนี้ ได้แก่ บริดจ์

**3. เน็ตเวิร์ค เลเยอร์** ( Network layer ) ทำหน้าที่ควบคุมการสื่อสารระหว่าง ต้นทาง กับปลายทาง (End to End) ซึ่งในระหว่างทางจะมีเครือข่ายอยู่หรือไม่ก็ได้ รวมทั้งการหาเส้นทางที่เหมาะสมในการเดินทางของข้อมูลจากต้นทางไปยังปลายทาง (ทำโดยเราท์เตอร์) มาตรฐานในระดับนี้ ได้แก่ X.25, ไอพี, เอสพีเอ็กซ์ เป็นต้น อุปกรณ์ในเครือข่ายที่ทำงานในชั้นนี้ เช่น เราท์เตอร์

4. **ทรานสปอร์ต เลเยอร์** ( Transport layer ) ทำหน้าที่ควบคุมการสื่อสารระหว่างต้นทาง และปลายทางให้สามารถได้รับข้อมูลที่ถูกต้องปราศจากข้อผิดพลาด (Error free) มาตรฐานใน ระดับนี้ ได้แก่ TCP, IPX เป็นต้น

5. **เซสชั่น เลเยอร์** ( Session layer ) ทำหน้าที่ควบคุมการจัดการจราจรในการสื่อสาร เช่น การหยุดส่งข้อมูลชั่วคราวเมื่อฝ่ายรับรับข้อมูลไม่ทัน หรือประมวลผลข้อมูลนั้นยังไม่เสร็จ เป็นต้น ซึ่งโดยปรกติจะจัดการโดยโปรแกรมประยุกต์เอง

6. **พรีเซนเตชั่น เลเยอร์** ( Presentation layer ) ทำหน้าที่ควบคุมการแสดงผลข้อมูล การแทนค่าข้อมูล การให้ความหมายข้อมูล เช่น จะให้ข้อมูลชุดนี้แทนรหัส ASCII หรือ EBCDIC, การตีความกลุ่มของข้อมูลว่าเป็น เลขจำนวนเต็ม หรือ ตัวอักษร หรือ จำนวนจริง เป็นต้น ซึ่งโดยปรกติจะจัดการโดยโปรแกรมประยุกต์เอง

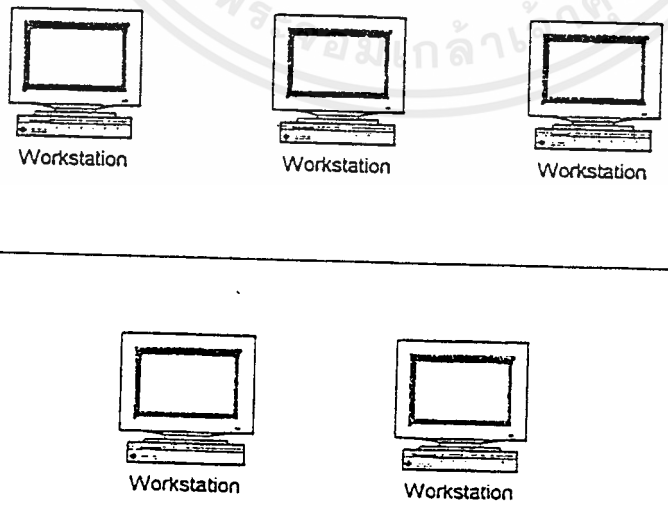
7. **แอพพลิเคชั่น เลเยอร์** ( Application layer ) เป็นระดับโปรแกรมประยุกต์ เช่น การส่งข้อความร้องขอข้อมูล การส่งข้อมูลตอบกลับ เป็นต้น

## 2.2 อีเทอร์เนต

อีเทอร์เนตได้ถูกนำมาใช้อย่างกว้างขวางในการทำโปรโตคอลระดับล่าง ซึ่งมีประวัติความเป็นมาและมีผู้ใช้และ การพัฒนามากมายซึ่งในส่วนนี้ จะกล่าวถึงรายละเอียดของอีเทอร์เนต

### 2.2.1 ประวัติการพัฒนา

อีเทอร์เนตเริ่มต้นในปี ค.ศ. 1970 เกิดจากการค้นคว้าและวิจัยของ Palo Alto Research Center ซึ่งเป็นส่วนหนึ่งของบริษัท Xerox ต้องการให้สามารถใช้งานสื่อสารระหว่างคอมพิวเตอร์ได้อย่างกว้างขวางและรวดเร็วจึงต้องใช้ความเร็วสูงในการส่งข้อมูล ซึ่งนี้คือจุดกำเนิดของ อีเทอร์เนต




*รูปที่ 2.3 แสดงการเชื่อมต่อเครื่องคอมพิวเตอร์ในระบบเครือข่าย*

วิธีการที่ใช้คือ ฟังสื่อที่ใช้ในการสื่อสารว่าว่างหรือไม่ แล้วจึงส่งข้อมูลลงไป ถ้าส่งไม่ได้เนื่องจากมีการใช้งานอยู่ก็จะรอช่วงเวลาหนึ่ง แล้วจึงส่งข้อมูลอีกที แต่อาจจะมีโอกาสที่ข้อมูลชนกันอีกได้เนื่องจากส่งข้อมูลพร้อมกัน จึงต้องมีวิธีการในการป้องกันการส่งพร้อมกันซึ่งเรียกว่า แบ็คออฟ อัลกอริทึม (backoff algorithm) ซึ่งจะกล่าวในภายหลัง ในตอนเริ่มแรกมีความเร็วเพียง 2.67 เมกกะบิตต่อวินาที ซึ่งสร้างในปี ค.ศ. 1973 ถึง 1975 ซึ่งรุ่นที่ออกมานั้นยังเป็นรุ่นทดสอบอยู่ ซึ่งมีข้อกำหนดที่ความเร็วและจำนวนสเตชั่นที่เชื่อมต่อ (station)

ในปีค.ศ. 1980 เด็ค ,อินเทลดอร์ปเปอเรชัน และ ซีรอก ร่วมกันกำหนดมาตรฐานอีเทอร์เนตซึ่งกำหนดเป็นรุ่นที่ 1 โดยมีคุณสมบัติดังนี้

- 10 เมกกะบิตต่อวินาที
- ความยาวสูงสุด 2.8 กิโลเมตร
- จำนวนสเตชันสูงสุด 1024 เครื่อง
- ใช้หลักการส่งแบบเบสแบนด์ (baseband)
- โทโพโลจีแบบบัส (bus)
- ขนาดเฟรมเปลี่ยนแปลงได้

### 2.2.2 หลักการทำงาน

อีเทอร์เนตใช้หลักการของ CSMA/CD ซึ่งทุกเวิร์คสเตชัน (workstation) ใช้สายสื่อระบบร่วมกันตามรูป เพราะว่าทุกเครื่องสามารถส่งข้อมูลได้ในเวลาเดียวกันบนสาย ซึ่งเป็นสาเหตุให้สัญญาณข้อมูลในสายเกิดการชนกันของข้อมูล ดังนั้นจะต้องส่งข้อมูลนั้นใหม่หมด

#### 2.2.2.1 ข้อดีและข้อเสียของอีเทอร์เนต

**ข้อดี**

**ง่ายในการติดตั้ง** : คอมพิวเตอร์สามารถเชื่อมต่อกับเซกเมนต์ใด ๆ ได้ง่ายเพียงแต่ใช้ ที-คอนเนคเตอร์ (T - connector) หรือตัวรับส่ง (Transceiver) อื่น ๆ

**เป็นวิทยาการที่ใช้กันอย่างแพร่หลาย** : มีการใช้งานในอีเทอร์เนตมาเป็นระยะเวลานาน มีเครื่องมือที่ใช้ในการเชื่อมต่อมากมายง่ายในการเลือกใช้

**ราคาอุปกรณ์ที่ใช้ไม่แพงมาก** : อุปกรณ์ที่จำเป็นต่อการเชื่อมต่อเช่น การ์ดอินเทอร์เฟสนั้นมีราคาไม่แพงมากนัก ทำให้ใช้กันอย่างแพร่หลาย

**รูปแบบการใช้งานมีอย่างกว้างขวาง** : อีเทอร์เนตสามารถใช้ชนิดของสายได้หลายแบบ เช่น 10base5 , 10base2 , 10baseT , Fiber optic

**ข้อเสีย**

- **เมื่อมีการใช้งานเพิ่มขึ้นประสิทธิภาพการใช้งานจะลดลง** : หลักการของ CSMA/CD เมื่อมีการใช้งานของเครือข่ายเพิ่มขึ้น ประสิทธิภาพจะลดลงเพราะเนื่องจากการชนกันของข้อมูลยิ่งมีมากขึ้น ถ้ามีการใช้งานเพิ่มขึ้นเพราะโอกาสที่จะส่งข้อมูลพร้อมกันในสายสื่อระบบยิ่งมีมากขึ้น
- **ยากในการตรวจสอบปัญหา** : อีเทอร์เนตยากที่จะตรวจสอบปัญหาที่เกิดขึ้นกับสายเพราะถ้าสายขาดแล้วจะทำให้ระบบแลนทั้งเซกเมนต์จะไม่ทำงาน

ก่อนที่จะใช้งานบนเครือข่ายแบบอีเทอร์เนตจะต้องเข้าใจถึงการทำงานของ CSMA/CD ซึ่งจะช่วยในการเข้าใจถึง สถิติ ข้อผิดพลาด และการใช้งาน ซึ่งจะกล่าวในส่วนต่าง ๆ ของปริญญานิพนธ์ฉบับนี้

**2.2.2.2 การทำงานของ CSMA/CD ส่วนการส่งข้อมูล**

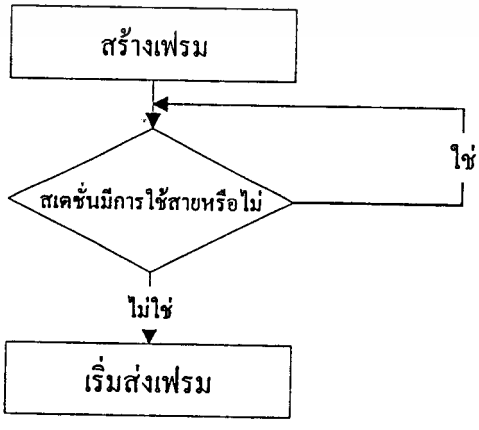
เนื่องจาก CSMA/CD ใช้สายสื่อระบบร่วมกัน ดังนั้นจึงมีข้อกำหนดในการส่งข้อมูล อย่างไรก็ตามก็ยังมีโอกาสที่จะทำให้เกิดการส่งข้อมูลพร้อมกันได้ ( CSMA/CD โปรโตคอลคล้ายกับการพูดคุยทางโทรศัพท์ ในขณะที่คนหนึ่งกำลังพูดคุยกันอยู่ในสาย ถ้าอีกคนหนึ่งพูดขึ้นมาพร้อมกันด้วยจะไม่ทราบข้อมูลที่อีกฝ่ายส่งมาได้ ต้องพูดคุยกันใหม่อีกครั้ง )

**2.2.2.2.1 ขั้นตอนในการทำงานในการส่งข้อมูลบน CSMA/CD**

มีด้วยกันทั้งหมด 4 ขั้นตอนดังนี้

**ขั้นที่ 1 ฟังก่อนที่จะส่ง**

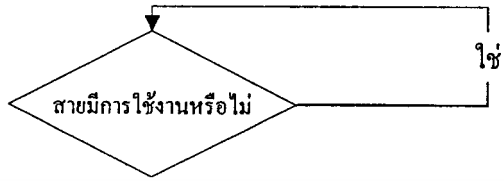
สเตชันจะเฝ้าดูว่าสายที่ส่งนั้นมีสัญญานแคร์เรีย (carrier) หรือไม่ตามรูปที่ 2.4 สัญญานนี้วัดโดยค่าระดับความต่างศักย์ซึ่งบ่งบอกถึงการใช้งานของสายนั้น ถ้าสเตชันไม่พบ แคร์เรียออน (carrier on) จะหมายความว่าสายนั้นว่างและพร้อมที่จะส่ง (เหมือนกับการที่เรายกหูโทรศัพท์เพื่อดูว่าสายว่างหรือไม่) ถ้าสายไม่ว่าง (แคร์เรียออน) เมื่อสเตชันกำลังจะส่ง แพ็กเกจที่ส่งนั้นจะชนกัน




รูปที่ 2.4 สเตชั่นเฝ้าดูการใช้งานสาย

**ขั้นที่ 2 รอถ้าสายไม่ว่าง**

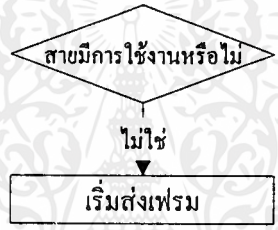
เพื่อป้องกันการชนกัน สเตชันจะรอถ้าสายนั้นถูกใช้งานอยู่ซึ่งแสดงตามรูป 2.5 ซึ่งโดยปกติแล้ว การ์ดอินเทอร์เฟสยังไม่ส่งถ้าสายไม่ว่าง (เปรียบกับโทรศัพท์แล้วเหมือนกับว่า เมื่ออีกคนกำลังจะพูดอยู่ ต้องรอให้พูดเสร็จก่อนจึงค่อยพูดต่อ) Deferal time คือช่วงเวลาที่สเตชันจะต้องรอก่อนที่จะพยายามส่งใหม่อีกครั้ง




รูปที่ 2.5 สเตชั่นรอเวลาถ้าสายไม่ว่าง

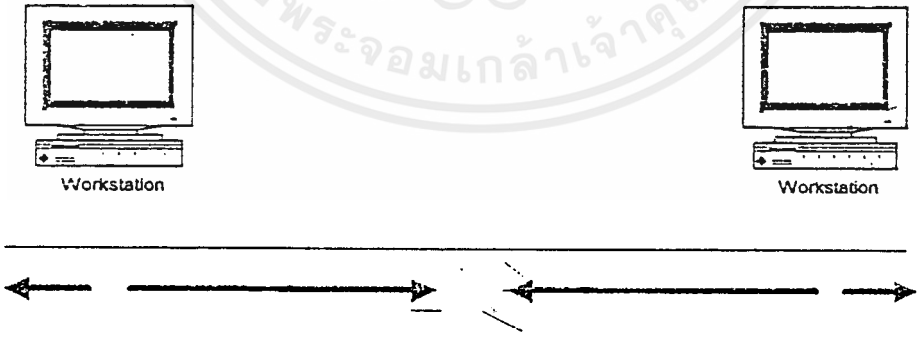
**ขั้นที่ 3 ส่ง และ รอว่าแพ็กแกจที่ส่งไปนั้นชนกันหรือไม่**

เมื่อสายว่างอย่างน้อยต้องใช้เวลา 9.6 ไมโครวินาที ถึงจะส่งข้อมูลตามรูป แล้วจึงส่งแพ็กแกจลงไปในสายสื่อระบบ



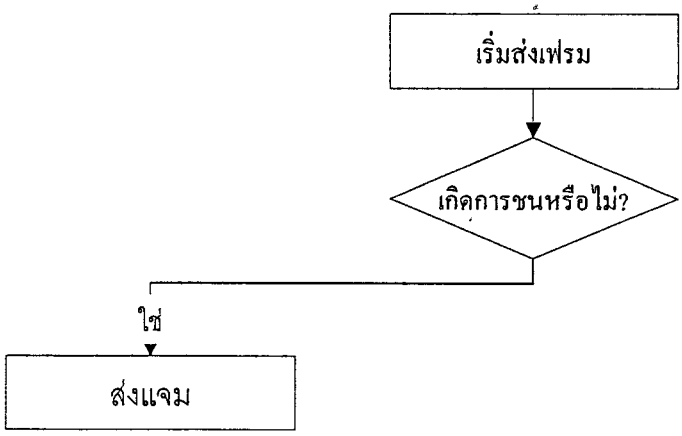

รูปที่ 2.6 ถ้าสายว่างสเตชั่นจะเริ่มส่งแพ็กเกจ

ถ้าสเตชันอื่นบนเซกเมนท์ส่งแพ็กเกจพร้อมกัน จะชนกันได้ตามรูปที่ 2.7 แสดงถึงการชนกันของแพ็กเกจ (ถ้าคุณพูดพร้อมกันในโทรศัพท์ก็จะคุยกันไม่รู้เรื่อง ) หลังจากส่งไปแล้วสเตชันก็จะทดสอบว่าสายมีการชนกันหรือไม่ ซึ่งการชนกันจะตรวจสอบได้โดยสัญญาณที่เกิดบนสายซึ่งจะเท่ากันหรือมากกว่าสัญญาณที่เกิดจากการส่งข้อมูล หรือมากกว่านั้นพร้อมกัน




*รูปที่ 2.7 เมื่อมีการชนกันของแพ็กเกจในสื่อ*

ถ้าเกิดการชนขึ้นแต่สเตชันอื่นไม่พบสัญญาณว่าชนกัน อาจจะพยายามส่ง แต่ก็จะเกิดการชนกันอีก เพื่อป้องกันการเกิดเหตุการณ์แบบนี้ สเตชันจะต้องทำให้ทุกสเตชันจะต้องรู้ถึงว่าสายโดยส่งแจม (jam) ดังแสดงตามรูปที่ ( แจม คือการส่งอย่าง 32 บิตที่ไม่เท่ากับค่า ซีอาร์ซี ของการส่งครั้งก่อนหน้านี้) สเตชันจะเพิ่มค่าพยายามในการส่งอีก 1

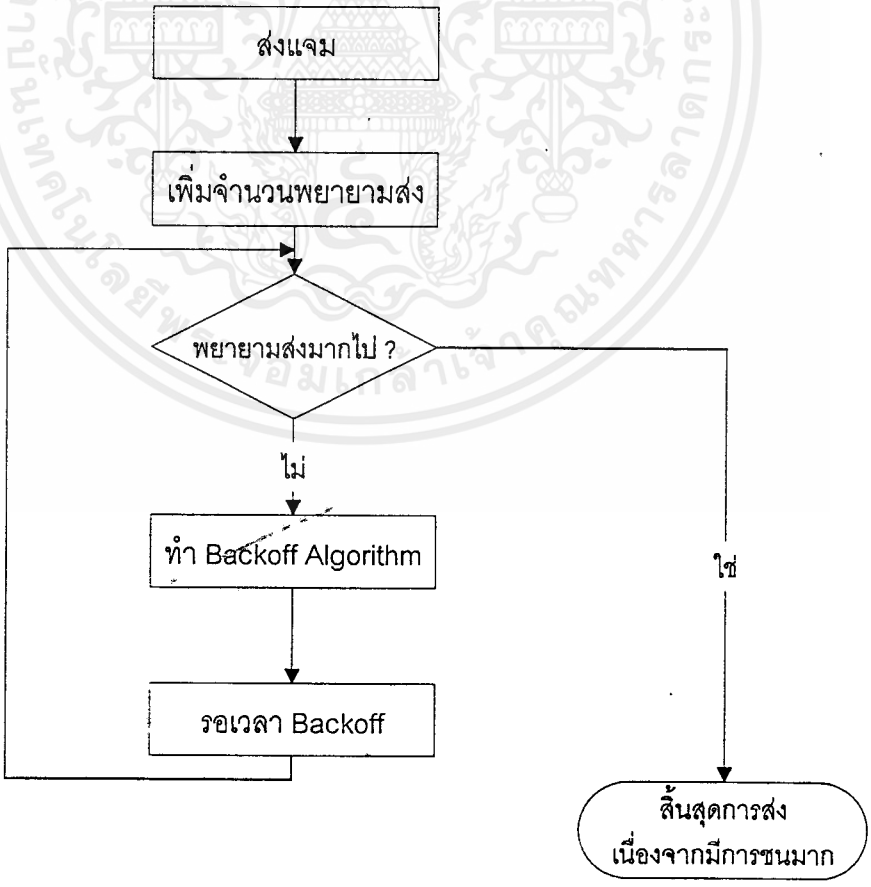



รูปที่ 2.8 ถ้ามีการชนเกิดขึ้นสเตชั่นจะส่งแจม

**ขั้นที่ 4 รอก่อนจะส่งใหม่**

ถ้าเราส่งทันทีหลังจากชนกันจะทำให้เกิดการชนครั้งที่สอง จำเป็นอย่างยิ่งที่จะกำหนดเวลาสุ่มเพื่อที่จะรอไม่ให้ชนกันอีก

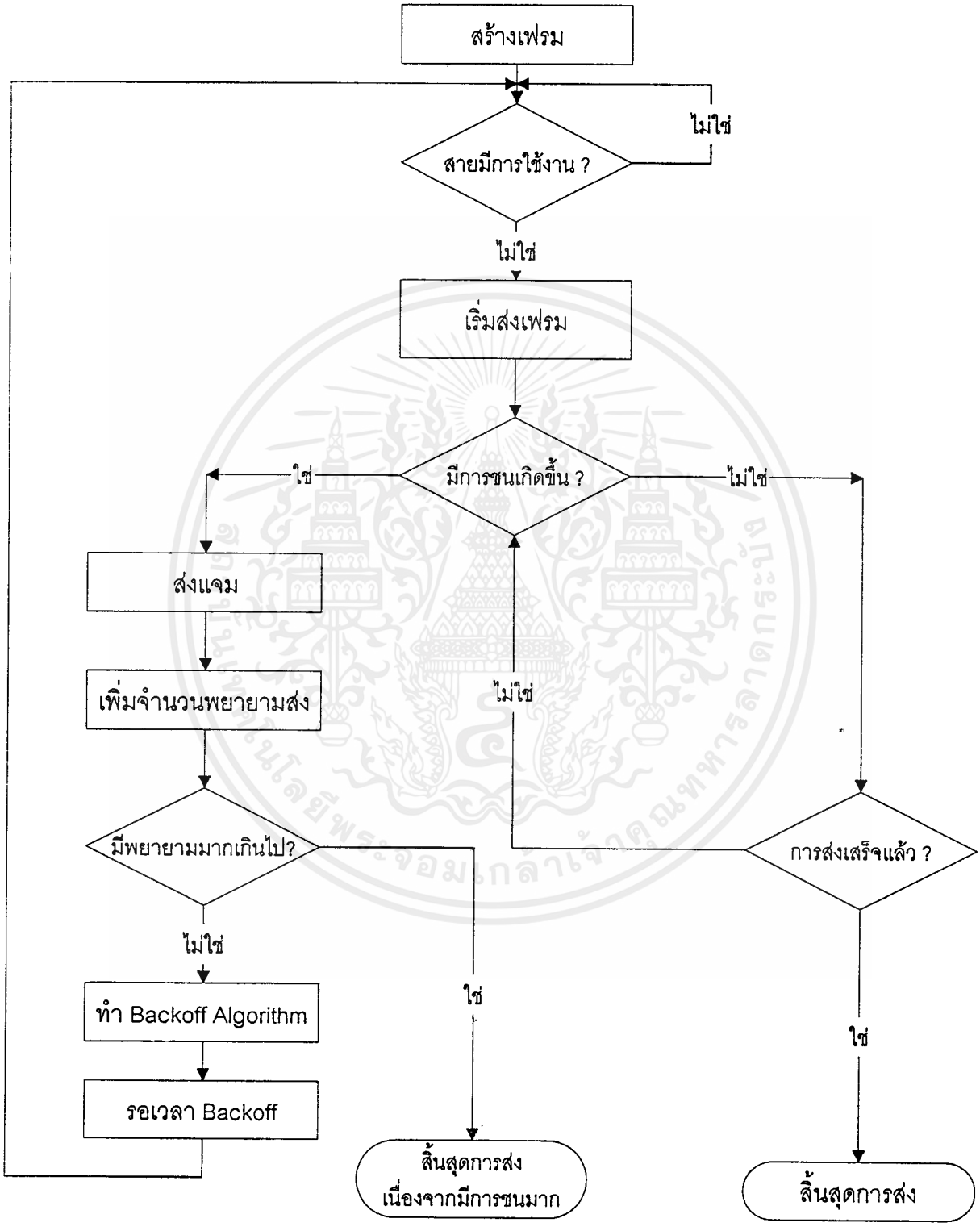
เพื่ออธิบายถึงการส่งข้อมูลใหม่อีกครั้ง สเตชันจะทำวิธีการที่เรียกว่า แบ็คออฟอัลกอริทึม ซึ่งกำหนดเวลาสุ่มที่จะใช้เพื่อการรอก่อนที่จะส่งแพ็กเกจอีกครั้งหนึ่ง ตามรูปที่ เพื่อลดการชนกันอีกครั้งหนึ่ง (ยกตัวอย่าง การคุยกันพร้อมกันก็จะต้องหยุดกันทั้งสองฝ่ายแล้วจึงมีฝ่ายหนึ่งค่อยพูดขึ้นมาอีกครั้งหนึ่ง)




รูปที่ 2.9 สเตชั่นใช้วิธีการแบ็คออฟ เพื่อที่จะใช้ในการส่งแพ็กเกจอีกครั้ง

**ขั้นที่ 5 ส่งอีกครั้งหรือหยุดการส่ง**

ถ้ามีการส่งแพ็กเกจไปเรื่อย ๆ แต่ไม่สามารถส่งไปในเครือข่ายได้อาจเป็นไปได้ที่มีปัญหาขึ้นในเครือข่าย จึงต้องกำหนดจำนวนครั้งในการพยายามที่จะส่งแพ็กเกจ โดยทั่วไปจะกำหนดไม่เกิน16 ครั้ง



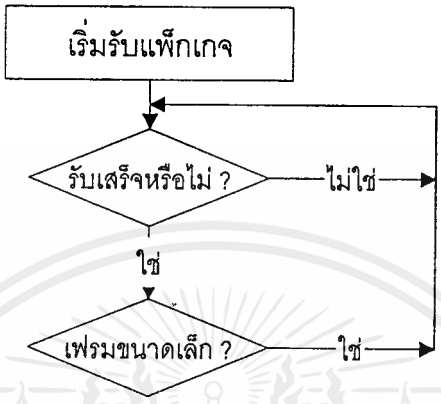

**รูปที่ 2.10 ลำดับขั้นการส่งแพ็กเกจ**

2.2.2.2.2 ขั้นตอนการทำงานในการรับข้อมูลบน CSMA/CD

มีด้วยกันทั้งหมด 4 ขั้นตอน ดังนี้

**ขั้นที่ 1 ตรวจสอบแพ็กเกจที่ได้รับ**

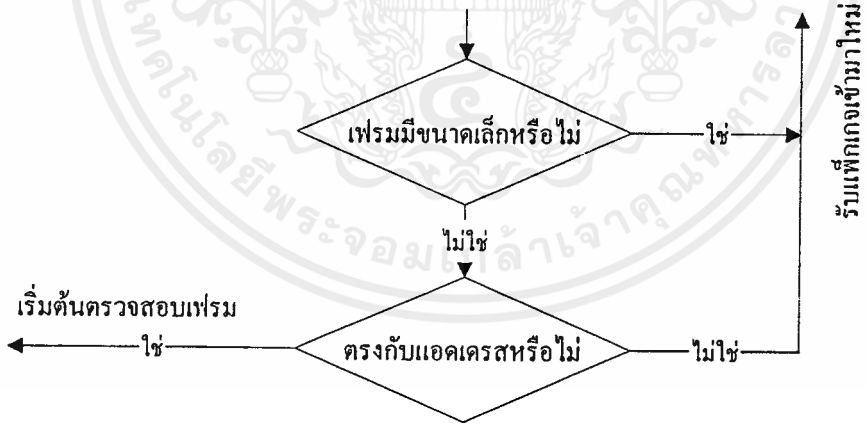
บนอีเทอร์เนตทุกสเตชันจะดูแพ็กเกจที่อยู่บนสาย เพื่อที่จะดูถึงแอดเดรสที่ส่งมายังสเตชันนั้นๆ และต้องตรวจสอบว่ามีขนาดถูกต้องหรือไม่ (อย่างน้อย 64 ไบต์) และมีครบถ้วนไม่แฟรกเมนท์เนื่องจากการชน ซึ่งดูได้จากรูปที่ 2.11




รูปที่ 2.11 เมื่อตรวจสอบแพ็กเกจสเตชั่นจะมองหาแฟรกเมนท์

**ขั้นที่ 2 ตรวจสอบว่าแอดเดรสปลายทาง**

หลังจากตรวจสอบแอดเดรสแล้วว่าไม่แฟรกเมนท์ ก็จะต่รวจสอบว่าแอดเดรสปลายทางของแพ็กเกจที่ตรวจสอบนั้นว่าเป็น บรอดคาส หรือ มัลติคาสหรือไม่ ตามรูปที่ 2.12




รูปที่ 2.12 สเตชันตรวจสอบที่อยู่ปลายทาง

**ขั้นที่ 3 ตรวจสอบถึงความถูกต้องของแพ็กเกจ**

ในจุดนี้สเตชันที่รับจะรู้ว่าแพ็กเกจไม่แฟรกเมนท์ และ เป็นแอดเดรสที่รับได้ แต่จะไม่รู้ว่าแพ็กเกจอาจจะเสียหายเนื่องจากการส่งข้อมูลในสายหรือไม่ เพื่อที่จะป้องกันการรับแพ็กเกจที่เสียหายนี้ สเตชันที่รับจะต้องตรวจสอบถึงคุณสมบัติของแพ็กเกจตามรูปที่ 2.13

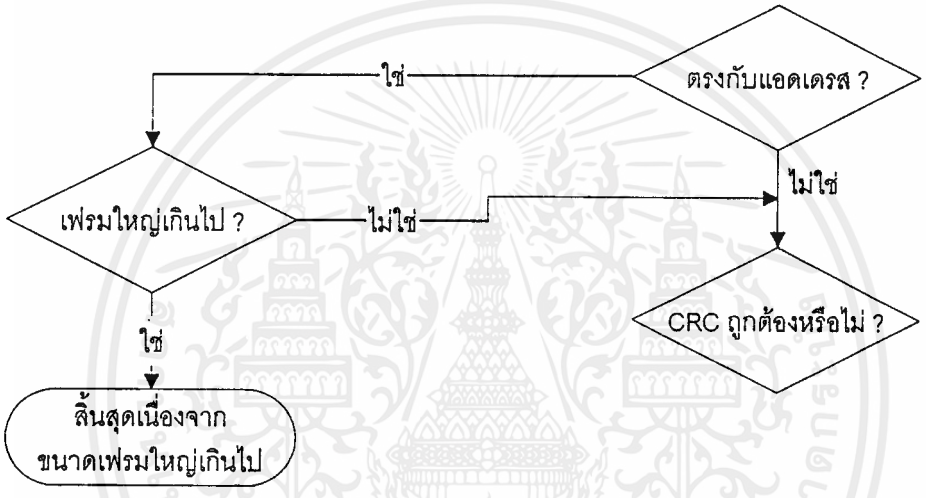
อันดับแรกถ้าจะต้องตรวจสอบขนาดของแพ็กเกจ ถ้าเฟรมขนาดใหญ่เกินกว่า 1518 ไบต์ จะถือว่าเป็น โอเวอร์ไซส์เฟรม (oversized frame) ซึ่งจะตรวจสอบในแพ็กเกจไดร์วเวอร์

บางทีอาจจะมีการสลับบิตได้ จาก 1 เป็น 0 หรือกลับกัน (ซึ่งทำให้แพ็กเกจเสียหาย) เพราะขณะที่ส่งไปบนสายสื่อระบบอาจจะมีการรบกวนของสภาพแวดล้อมซึ่งเป็นสาเหตุให้เกิดเหตุการณ์เช่นนี้ได้ ซึ่งจะต้องตรวจสอบ ซีอาร์ซี (CRC: Cyclical Redundancy Check) ถ้าตรวจสอบซีอาร์ซี แล้วไม่ผ่านจะตรวจสอบเฟรมนั้นต่อว่า อไลน์เมนท์ (alignment) ถูกต้องหรือไม่

แพ็กเกจ มิส อไลน์ (misaligned) คือแพ็กเกจที่ไม่จบลงด้วยจำนวนเท่าของ 8 บิต เช่น แพ็กเกจขนาด 72 ไบต์ 3 บิต อาจเป็นไปได้ที่เป็น 72 ไบต์ หรือ 73 ไบต์

เมื่อตรวจสอบว่าแฟรกเมนท์ ยาวเกินไป มีซีอาร์ซีไม่ถูกต้อง จัดเรียงผิดพลาด (มิส อไลน์) หลังจากนั้นก็ตรวจสอบว่าแพ็กเกจมีขนาดเล็กเกินไปหรือไม่ โดยพิจารณาจากขนาดว่าอย่างน้อยต้องมีขนาดมากกว่า 64 ไบต์

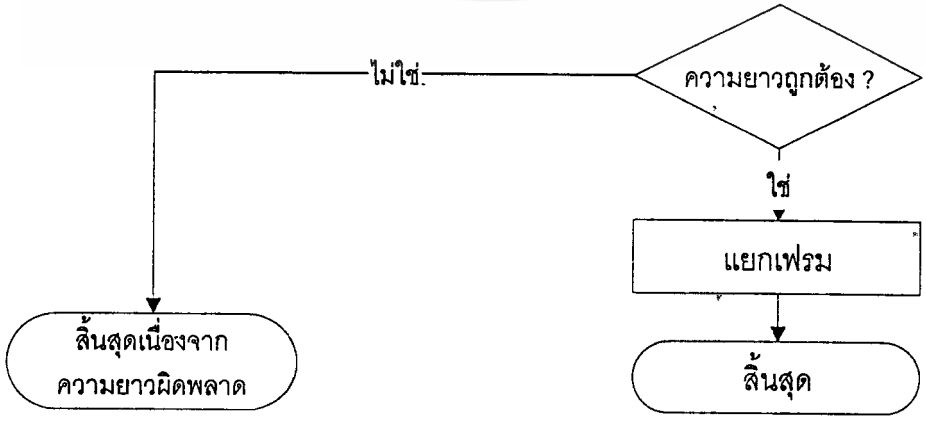
ตามรูปที่ 2.15 อธิบายถึงขั้นตอนการรับแพ็กเกจของสเตชันบนเครือข่ายแบบ CSMA/CD



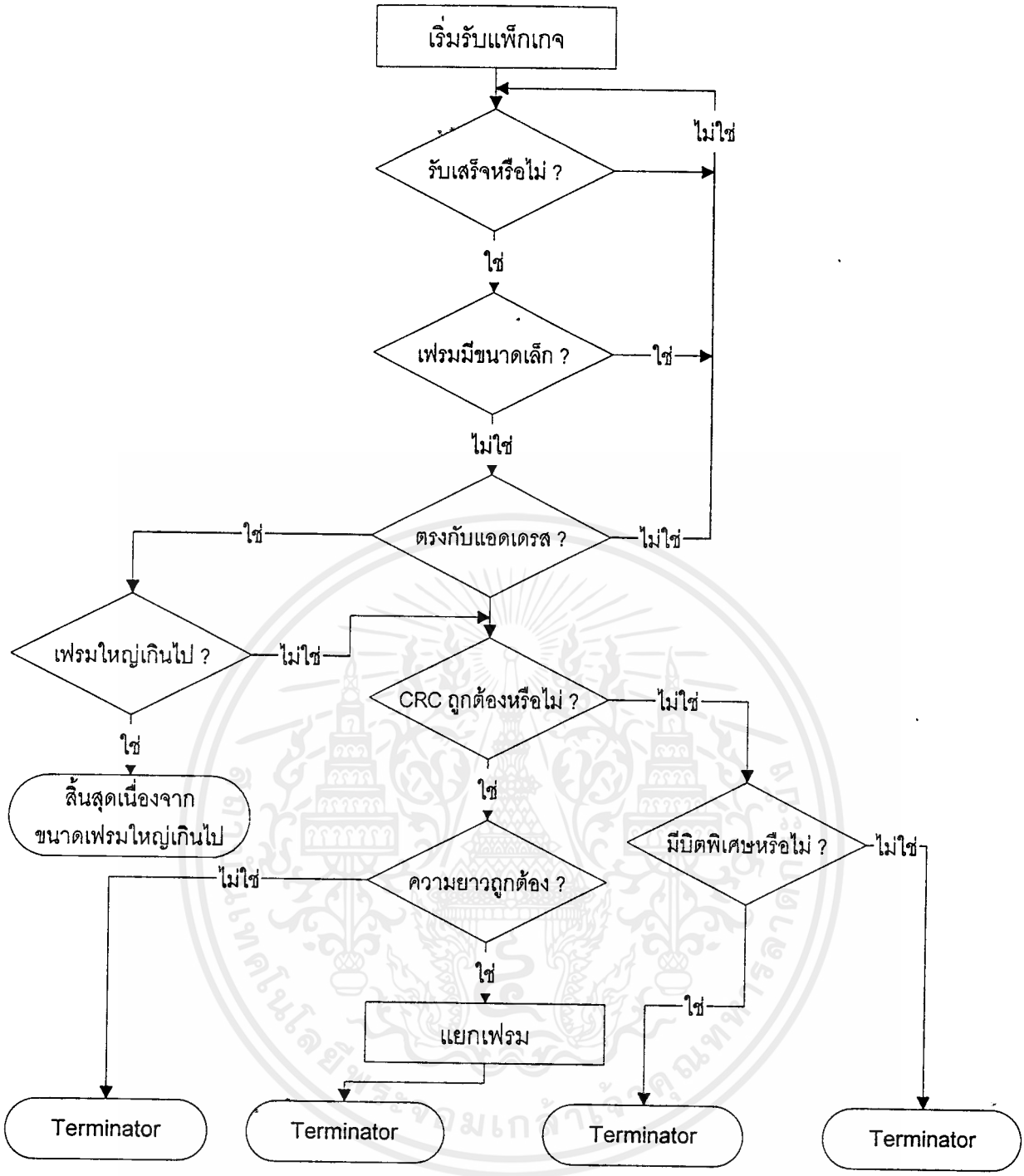

รูปที่ 2.13 แพ็กเกจถูกตรวจสอบสำหรับความถูกต้อง

**ขั้นที่ 4 โปรเซสแพ็กเกจ**

ถ้าแพ็กเกจผ่านขั้นตอนการตรวจสอบแต่ละขั้นแล้ว ตามรูปที่ 2.13 จะถือว่าแพ็กเกจนั้นถูกต้องในรูปแบบ ขนาด ถ้าสเตชันยังมีปัญหาอีกให้พิจารณาถึงข้างในแพ็กเกจเพื่อที่จะหาปัญหาที่เกิดขึ้น บางทีอาจเกิดขึ้นในโปรโตคอลเลเยอร์อื่นๆ ก็เป็นได้




รูปที่ 2.14 แพ็กเกจที่ถูกต้องจะถูกจัดการต่อ




รูปที่ 2.15 ลำดับขั้นการรับแพ็กเกจ

### 2.2.3 ส่วนประกอบของอีเทอร์เนตเฟรม

อีเทอร์เนตเฟรมมีขนาดประมาณ 46 และ 1500 ไบต์ เฟรมน้อยกว่า 60 ไบต์ในส่วนข้อมูลจะเรียกว่า รั๊น (runt) เฟรม รูปที่เป็นตัวอย่างของอีเทอร์เนตเฟรม

| Preamble | Destination Address | Source Address | Type | Data | FCS |
|---|---|---|---|---|---|

*รูปที่ 2.16 โครงสร้างเฟรมอีเทอร์เนต*

**โครงสร้างเฟรม**

เฟรมนี้ประกอบด้วย 6 ฟิลด์ด้วยกันทุกฟิลด์มีหน้าที่และขนาดเฉพาะ ซึ่งมีรายละเอียดดังนี้

**พรีแอมเบิล** (Preamble)

เป็นเลขลำดับ 64 บิตที่ฟิสิคัลเลเยอร์ ใช้เพื่อการสร้างสัญญาณพร้อม (synchronization signal) ระหว่างวงจรที่เชื่อมต่อกับสื่อ

**ที่อยู่ปลายทาง** (Destination Address)

เป็นเลข 48 บิต ฮาร์ดแวร์แอดเดรส ซึ่งเรียกว่า อีเทอร์เนตแอดเดรส หรือแมคแอดเดรส ( ทุกๆ การ์ดอินเทอร์เฟสจะมีเลข 48 บิต ที่บ่งบอกถึง ความแตกต่างของการ์ดอินเตอร์เฟสนั้น ) ของสเตชันที่ต้องการส่งไป

**ที่อยู่ต้นทาง** (Source address)

เป็นเลข 48 บิต ฮาร์ดแวร์แอดเดรส ของผู้ส่ง

**ไทป์** (TYPE)

เป็นเลข 2 ไบต์ใช้เพื่อบ่งบอกชนิดโปรโตคอลถ้ามีการใช้งานหลายโปรโตคอลในระดับบนสื่อเดียวกัน จะใช้เลขนี้บ่งบอกถึงโปรโตคอลในระดับบน

**ข้อมูล** (DATA)

เป็นข้อมูลขนาดระหว่าง 4 ถึง 1500 ไบต์

**เอฟซีเอส**(FCS : FRAME CHECK SEQUENCE)

เป็นเลขขนาด 32 บิตที่คำนวณจาก ซีอาร์ซีทุก ๆฟิลด์ยกเว้นฟิลด์ตัวเอง

ในสภาพแวดล้อมทั่วไป ในเครือข่ายจะมีโครงสร้างเฟรมของอีเทอร์เนต ที่ใช้พื้นฐานโครงสร้างของโครงสร้างเฟรมอีเทอร์เนต เพียงแต่แตกต่างกันตรงส่วนการใช้งานของแต่แบบไม่เหมือนกันซึ่งมีด้วยกันดังนี้

### 2.2.3.1 อีเทอร์เนต 802.3

อีเทอร์เนต 802.3 เฟรมนั้นคล้าย ๆ กับอีเทอร์เนต II แต่ไม่เหมือนกันตรงฟิลด์ต่าง ๆดูได้ตามรูปที่ 2.17 ซึ่งมีหน้าที่และขนาดเฉพาะซึ่งมีรายละเอียดดังนี้

| Preamble | Destination Address | Source Address | Length | Data | FCS |
|---|---|---|---|---|---|

*รูปที่ 2.17 โครงสร้างเฟรมอีเทอร์เนต 802.3*

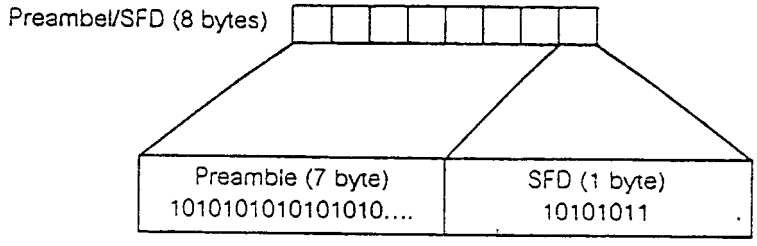
**โครงสร้างเฟรม**

**พรีเอมเบิ้ล**

เป็นเลขอันดับ 56 บิตหรือ 7 ไบต์ที่ฟิสิคัลเลเยอร์ใช้เพื่อการสร้างสัญญาณพร้อม (synchronization signal) ระหว่างวงจรที่เชื่อมต่อกับสื่อ

**เอสเอฟดี** (SFD : Stat frame delimiter)

เป็นเลขไบนารี่ (binary) 10101011 ที่ชี้ถึงจุดเริ่มต้นของเฟรมดังตัวอย่างตามรูปที่ 2.18




*รูปที่ 2.18 โครงสร้างฟิลด์เอสเอฟดี*

**หมายเลขปลายทาง**

เป็นเลข 48 บิต ฮาร์ดแวร์แอดเดรส ซึ่งเรียกว่าอีเทอร์เนตแอดเดรส หรือ แมคแอดเดรส (ทุกๆ การ์ดอินเทอร์เฟสจะมีเลข 48 บิต ที่บ่งบอกถึงความแตกต่างของการ์ดอินเทอร์เฟสนั้น) ของสเตชันที่ต้องการส่งไปแอดเดรส FFFFFFFFFFFF หมายถึงบรอดคาส

**หมายเลขต้นทาง**

เป็นเลข 48 บิต ฮาร์ดแวร์แอดเดรส ของผู้ส่งที่จะต้องไม่เป็น บรอดคาส จะต้องเป็นแอดเดรสของ เวิร์คสเตชัน เซิฟเวอร์ หรือ เราท์เตอร์

**ความยาว** (LENGTH)

เป็นเลข 2 ไบต์ใช้เพื่อบอกขนาดแพ็กแกจซึ่งค่านี้จะต้องน้อยกว่า 1500

**ข้อมูล**

เป็นข้อมูลขนาดระหว่าง 46 ถึง 1500 ไบต์

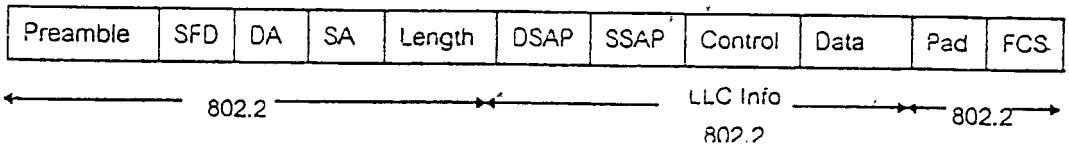
**แพดดิ้ง** (PADDING)

ฟิลด์นี้เปลี่ยนแปลงได้ ใช้เพื่อตรวจสอบแพ็กเกจให้มีขนาดตรงตามข้อกำหนด เช่นในอีเทอร์เนตต้องมีขนาดอย่างน้อย 64 ไบต์ ซึ่งถ้ามีการส่งข้อมูลไม่ถึงจะต้องเพิ่มแพดดิ้งเพื่อเข้าไปให้ครบทั้ง 64 ไบต์

<u>หมายเหตุ</u> ถ้าเฟรมถูกต้องและ ค่าความยาวมากกว่า 1500 จะหมายความว่า เป็นอีเทอร์เนตทูเฟรม และเป็นไทป์ฟิลด์

### 2.2.3.2 อีเทอร์เนต 802.2

เฟรมอีเทอร์เนต 802.2 กำหนด IEEE - compliant เนื่องจากมีข้อมูลทั้ง 802.2 ฟิลด์ และ 802.2 ฟิลด์ ซึ่ง 802.2 ฟิลด์กล่าวถึง LLC (Logical Link Control) เลเยอร์ภายในเฟรม ดูได้จากรูปที่




*รูปที่ 2.19 โครงสร้างเฟรมอีเทอร์เนต 802.2*

**โครงสร้างเฟรม**

**พรีเอมเบิ้ล : 8 ไบต์**

**ที่อยู่ปลายทาง : 6 ไบต์**

**ที่อยู่ต้นทาง : 6 ไบต์**

**ความยาว : 2 ไบต์**

**ข้อมูลและแพดดิ้ง : 46 - 1500 ไบต์**

**เอฟซีเอส : 4 ไบต์**

**ดีเอสเอพี** (DSAP : Destinationn Service Access Point)

เป็นเซอร์วิสแอคเซสพอย์นปลายทางของโฮสปลายทางซึ่งใช้ในเลเยอร์บน หรือ เน็ตเวิร์คเลเยอร์

**เอสเอสเอพี**(SSAP : Source Service Access Point)

เป็นเซอร์วิสแอคเซสพอย์นต้นทางของโฮสต้นทางซึ่งใช้ในเลเยอร์บน หรือ เน็ตเวิร์คเลเยอร์

**ควบคุม**(CONTROL)

กำหนดถึงการส่งแบบคอนเน็คชั่นเลสเซอร์วิส(Connectionless Service)

**2.2.3.3 อีเทอร์เนต สแนบ**

สแนบมาจากเฟรม อีเทอร์เนต 802.2 ดังแสดงตามรูป

| Preamble | SFD | DA | SA | Length | DSAP | SSAP | Control | OrganizationCode | Data | Pad | FCS |
|---|---|---|---|---|---|---|---|---|---|---|---|

*รูปที่ 2.20 โครงสร้างเฟรมอีเทอร์เนต สแนบ*

**พรีเอมเบิ้ล : 8 ไบต์**

**ที่อยู่ปลายทาง : 6 ไบต์**

**ที่อยู่ต้นทาง : 6 ไบต์**

**ความยาว : 2 ไบต์**

**ข้อมูลและแพดดิ้ง : 46 - 1500 ไบต์**

**ดีเอสเอพี เอสเอสเอพี และ คอนโทรลฟิลด์**

**เอฟซีเอส : 4 ไบต์**

**ดีเอสเอพี เอสเอสเอพี และ คอนโทรล**

ในอีเทอร์เนตสแนบค่าใน ดีเอสเอพี และ เอสเอสเอพี จะต้องเป็น 0xAA ซึ่ง 0xAA เป็นตัวบ่งบอกถึง เฟรมที่เป็นสแนบ

ออร์กาไนซ์เซชั่นโคด (Organization Code)

ฟิลด์นี้กำหนดให้อีเทอร์เนตไทป์ฟิลด์ซึ่งค่าเป็น 0x00-00-00 ในออร์กาไนซ์เซชั่นฟิลด์

**อีเทอร์เนตไทป์**

ฟิลด์นี้กำหนดถึงโปรโตคอลระดับบน (Upper layer protocol) ซึ่งมีค่าดังนี้

| | |
|---|---|
| ไอพี | 0x0800 |
| เออาร์พี | 0x0806 |
| อาร์เออาร์พี | 0x8305 |
| แอปเปิ้ลทอร์ค | 0x809B |
| แอปเปิ้ลทอร์ค เออาร์พี | 0x80F3 |
| เน็ตแวร์ ไอพีเอ็กซ์/เอสพีเอ็กซ์ | 0x8137 |

**2.2.3.4 อีเทอร์เนตทู**

อีเทอร์เนตทูเฟรมแตกต่างจาก 2 แบบที่กล่าวมาเนื่องจากจากไทป์ซึ่งตามหลังที่อยู่ปลายทาง แต่อีเทอร์เนต 802.3 อีเทอร์เนต 802.2 และ อีเทอร์เนตสแนบ จะเป็นฟิลด์ความยาวแทน ซึ่งแสดงตามรูป

| Preamble | Destination Address | Source Address | Type | Data | FCS |
|---|---|---|---|---|---|

*รูปที่ 2.21 โครงสร้างเฟรมอีเทอร์เนตทู*

**โครงสร้างเฟรม**

**พรีเอมเบิ้ล**

เป็นเลขลำดับ 64 บิตหรือ 8 ไบต์ที่ฟิสิคัลเลเยอร์ใช้เพื่อการสร้างสัญญานพร้อม (Synchronization signal) ระหว่างวงจรที่เชื่อมต่อกับสื่อ กำหนดด้วยค่าสลับกันระหว่าง 1 และ 0 ซึ่งมีด้วยกัน 7 ไบต์ ส่วนไบต์สุดท้ายเป็นเอสเอฟดี

**ไทป์**

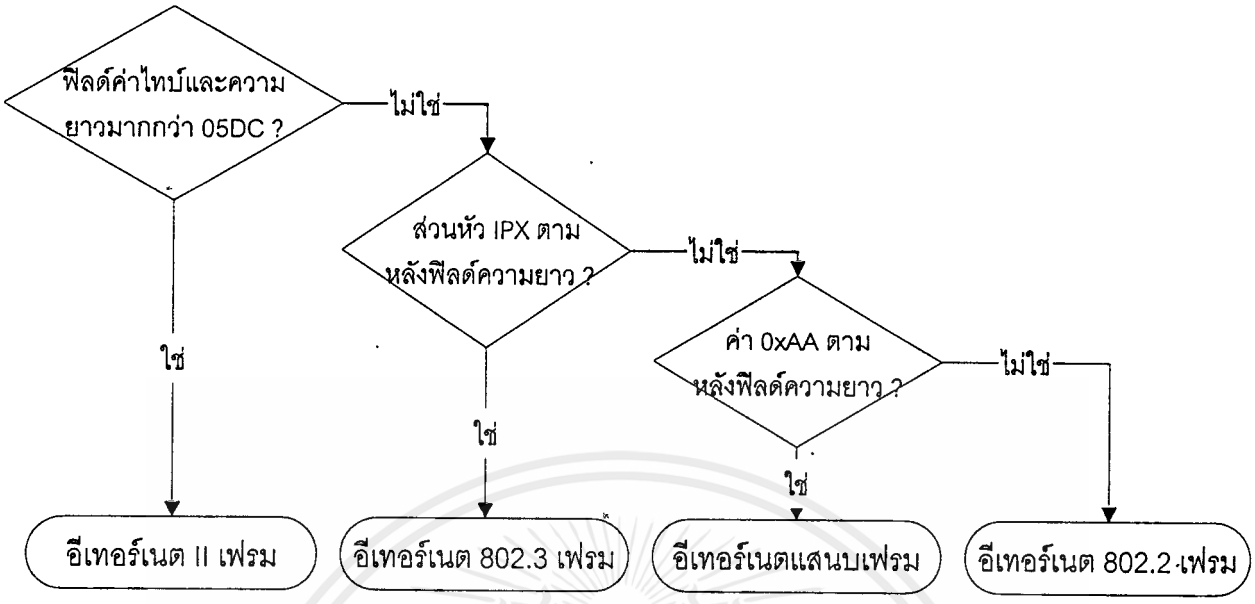
เป็นส่วนที่บอกถึงชนิดโปรโตคอลในระดับบนซึ่งค่าโปรโตคอลที่ใช้มีดังนี้

| | |
|---|---|
| ไอพี | 0x0800 |
| เออาร์พี | 0x0806 |
| อาร์เออาร์พี | 0x8305 |
| แอปเปิ้ลทอร์ค | 0x809B |
| แอปเปิ้ลทอร์ค เออาร์พี | 0x80F3 |
| เน็ตแวร์ ไอพีเอ็กซ์/เอสพีเอ็กซ์ | 0x8137 |

สังเกตุว่าค่าเหล่านี้จะคล้ายกับ อีเทอร์เนตแสนบไทป์ฟิลด์

### 2.2.4 วิธีการแยกชนิดเฟรม

จากรูปที่ 2.22 แสดงถึงวิธีการแยกประเภทเฟรมของอีเทอร์เนต




รูปที่ 2.22 แสดงลำดับขั้นตอนการแยกชนิดอีเทอร์เนตเฟรม

## 2.3 ทีซีพี/ไอพีโปรโตคอลซุท(TCP/IP Protocol suite)

กลุ่มของโปรโตคอลซุท ซึ่งจะให้บริการ ฟังก์ชั่น พื้นฐานสำหรับการใช้งาน ซึ่งรวม ไอพี ทีซีพี ยูดีพี สามารถนำมาใช้เป็นการส่งไฟล์ (file) เมล์ (mail) ค้นหาว่าใครใช้งานอยู่บ้าง ในเบื้องต้น ทีซีพี/ไอพี ถูกใช้อย่างมากในมินิคอมพิวเตอร์ (minicomputer) หรือ เมนเฟรม (mainframes) ดังนั้นความสัมพันธ์ของทีซีพี/ไอพี ในการให้บริการต่าง ๆ ได้แก่

-**ไฟล์ทรานสเฟอร์** (files transfer) ไฟล์ทรานสเฟอร์โปรโตคอล (FTP : Files transfer protocol) อนุญาตให้ผู้ใช้เครื่องคอมพิวเตอร์ใดก็ตาม สามารถรับ-ส่งไฟล์จากเครื่องคอมพิวเตอร์อื่นได้

-**รีโมทล็อกอิน**(remote login) เทลเน็ตโปรโตคอล (TELNET : Network terminal protocol) อนุญาตให้ผู้ใช้เข้าไปใช้เครื่องคอมพิวเตอร์ที่ใดก็ได้ในเน็ตเวิร์ค สามารถที่จะเริ่ม รีโมทเซสชั่น(remote session) โดยกำหนดคอมพิวเตอร์ที่จะเชื่อมต่อได้

-**อิเลคโทรนิกส์เมล์**(electronic mail) อนุญาตให้ผู้ใช้ส่งข้อความไปยังผู้ใช้คนอื่นในเน็ตเวิร์คได้ ซึ่งการให้บริการเหล่านี้เป็นการนำไปใช้งานของทีซีพี/ไอพี ซึ่งแสดงให้เห็นว่า ทีซีพี/ไอพี มีบทบาทเป็นพื้นฐานของเน็ตเวิร์ค ในปัจจุบัน การใช้งานของคอมพิวเตอร์ก็มีการเปลี่ยนแปลงไป แต่เดิมใช้เป็นลักษณะประมวลผลกลาง หรือ ใช้พวกเมนเฟรม ก็เปลี่ยนมาใช้การประมวลผลแบบกระจาย หรือใช้พวก ไคลน์เซิฟเวอร์(Client Server) แทน ซึ่งการจะทำงานเหล่านี้ก็ต้องใช้พื้นฐานของ ทีซีพี/ไอพี

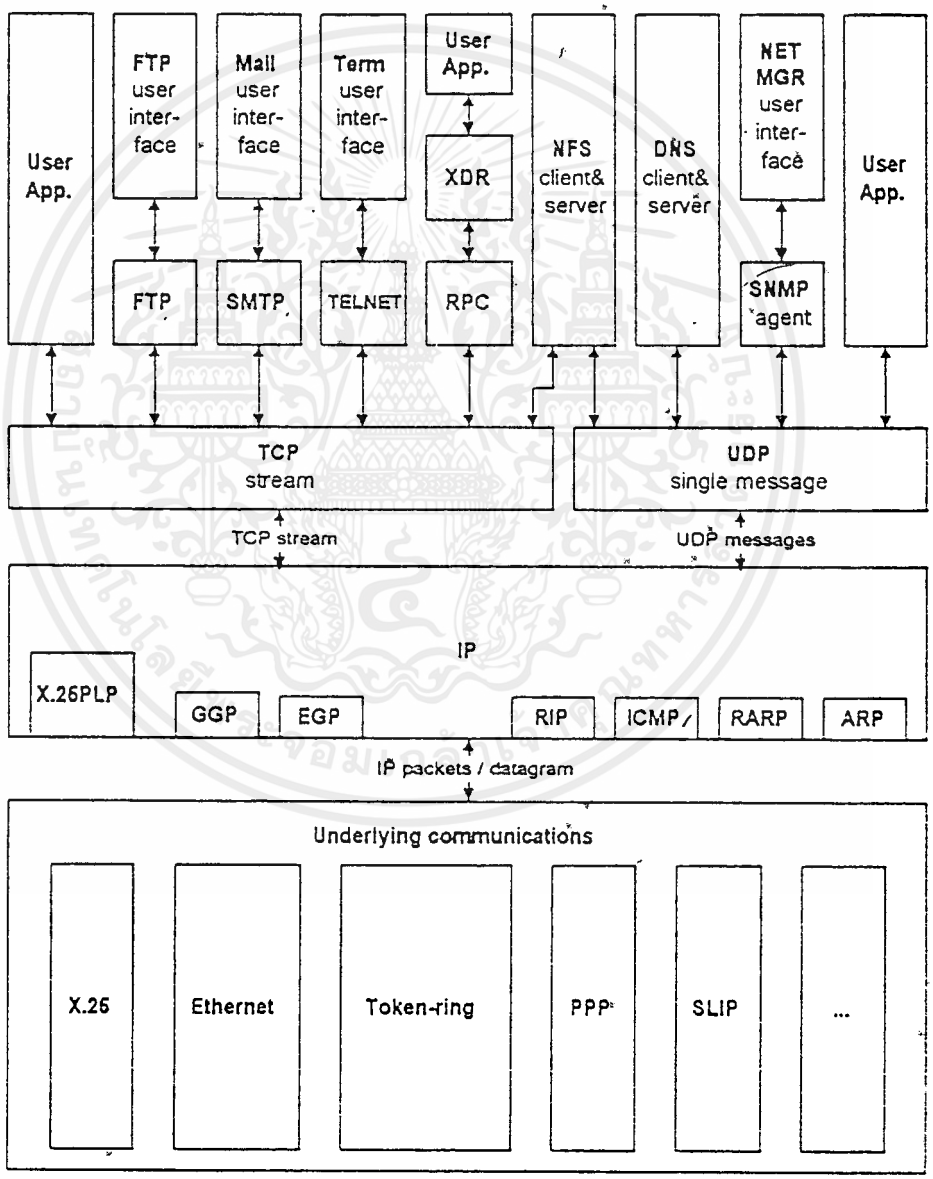
-**เน็ตเวิร์คไฟล์ซิสเต็ม** (NFS : network file system) เป็นการอนุญาตระบบให้เข้าสู่ข้อมูลจากคอมพิวเตอร์เครื่องอื่นได้โดยง่ายกว่าการใช้ เอฟทีพี ซึ่งมีประสิทธิภาพมากกว่า

-**การพิมพ์ปลายทาง**(remote printing) อนุญาตให้มีการใช้งานเครื่องพิมพ์ผ่านระบบเน็ตเวิร์คได้

-**การทำงานปลายทาง**(remote execution) ทำให้มีการทำงานข้ามระบบได้เพื่อประหยัดทรัพยากร โดยสามารถที่จะสั่งงานเครื่องใหญ่ๆ ได้จากเครื่องเล็กๆ

-**เนมเซิฟเวอร์**(name server) ในระบบใหญ่ๆ ที่มีหลายเครื่องการใช้ชื่อเป็นสื่อแทนตัวเลขจะเหมาะแก่การบริการคอมพิวเตอร์มากกว่า

-**เทอร์มินอลเซิฟเวอร์**(terminal server) ในการทำงานเราสามารถที่จะทำงานจากเทอร์มินอลเพื่อใช้ทรัพยากรร่วมกันได้




*รูปที่ 2.23 การใช้งานของโปรโตคอลต่างๆ*

ทีซีพี/ไอพี สร้างด้วยวิทยาการของคอนเน็คชั่นเลส(connectionless) ข้อมูลถูกส่งไปตามลำดับเป็นดาต้าแกรม(datagram) คือ กลุ่มข้อมูลที่ส่งไปหนึ่งแมสเซส และ ถูกส่งไปยังหลายระบบ เน็ตเวิร์ค หมายความว่า ข้อมูลได้ถูกแบ่งเป็นข้อมูลย่อยๆ หลายส่วนเพื่อทยอยส่งไป เนื่องจากว่า บางระบบ เน็ตเวิร์คไม่สามารถที่จะส่งข้อมูลขนาดใหญ่ได้ ต้องแบ่งข้อมูลออกเป็นส่วนย่อยๆ เพื่อส่งไปได้ และเมื่อส่งไปถึงปลายทางก็จะรวบรวมข้อมูลนั้นเป็นตัวเดิม

### 2.3.2 เลเยอร์ของ ทีซีพี/ไอพี

#### 2.3.2.1 อีเทอร์เนตเลเยอร์

ในระบบเน็ตเวิร์ค ส่วนใหญ่ใช้อีเทอร์เน็ต ดังนั้นในส่วนนี้จะอธิบายถึงส่วนหัวของอีเทอร์เนต ในการทำงานของอีเทอร์เนตจะต้องมีแอดเดรส (address) เป็นตัวกำหนดการติดต่อสื่อสาร ซึ่งหมายความว่าจะต้องมีหมายเลขที่ไม่ซ้ำกัน และ จะต้องไม่ให้ผู้ใช้ยุ่งยากในการตั้งค่าเหล่านี้ ดังนั้นหมายเลขเหล่านี้ถูกกำหนดมาจากโรงงานที่ทำ การ์ดอินเทอร์เฟส ซึ่งการกำหนดใช้เลขขนาด 48 บิต เมื่อข้อมูลถูกส่งออกไป จะเป็นการส่งแบบ บรอดคาสท์มีเดียม(broadcast medium) คือเมื่อ โฮสรับแพ็กเกจได้ก็จะตรวจสอบว่าเป็นแพ็กเกจที่ส่งมาถึงตนเองหรือไม่ ถ้าใช่ก็รับข้อมูลนั้น ซึ่งจะดูที่อีเทอร์เนตส่วนหัว ซึ่งทุกๆ อีเทอร์เนต แพ็กเกจจะมีส่วนหัวขนาด 14 อ็อกเตต ซึ่งจะอธิบายถึง ต้นทาง(source) และปลายทาง (destination ) และชนิด (type)

| | |
|---|---|
| Ethernet Destination address (first 32 bits) | |
| Ethernet dest (last 16 bits) | Ethernet source (first 16 bits) |
| Ethernet source address (last 32 bits) | |
| Type code | |
| Destination Address | |
| IP header, then TCP header, then your data | |
| ... | |
| end of your data | |
| Ethernet Checksum | |
| | |
| | |

*รูปที่ 2.24 โครงสร้างของอีเทอร์เนตโปรโตคอล*

ถ้าเราใช้ E แทน อีเทอร์เนตส่วนหัว ใช้ C แทน ผลรวมตรวจสอบอีเทอร์เนต ( ethernet checksum) ใช้ I แทนส่วนหัวไอพี T แทนหัวทีซีพี เราจะเห็นข้อมูลในรูปแบบดังนี้

EIT...C EIT...C EIT...C EIT...C

เมื่อได้รับแพ็กเกจแล้ว จะเอาส่วนหัวและ ผลรวมตรวจสอบออก เนื่องจากไม่ใช้อีกแล้ว และพิจารณาที่ไทป์โค้ด(type code) ถ้าเป็นโค้ด(code) ของไอพี ก็จะส่งดาต้าแกรมไปยังไอพี ในส่วนไอพีจะ

เอาไอพีส่วนหัวออก และดูตรงส่วน ไอพีโปรโตคอลฟิลด์ และถ้าเป็น ทีซีพี ก็จะส่งไปยังส่วนของทีซีพี ในส่วนนี้จะดูที่ หมายเลขลำดับ เพื่อที่จะรวบรวมข้อมูลเป็นข้อมูลเดิม

**2.3.2.2 ระดับชั้นไอพี**

ทีซีพีส่งดาต้าแกรมมายังชั้นไอพี ซึ่งจะบอกถึงอินเทอร์เนตแอดเดรสของเครื่องปลายทาง สังเกตว่าในชั้นไอพีนั้นไม่คำนึงถึงข้อมูลที่ส่งมาจากชั้นบน(ดาต้าแกรม) แม้กระทั่งส่วนหัวของทีซีพี งานหลักของไอพีคือ ค้นหาเส้นทางเพื่อที่จะส่งไปยังปลายทาง ในการที่จะให้เกตเวย์ (gateway) ส่งดาต้าแกรมจะต้องมีส่วนของส่วนหัวในดาต้าแกรมซึ่งในส่วนนี้จะบอกถึง อินเทอร์เนตแอดเดรส ต้นทางและปลายทาง (32 บิตแอดเดรส เช่น 161.246.6.71) และโปรโตคอลนัมเบอร์ (protocol number) และผลรวมตรวจสอบตำแหน่งต้นทางและปลายทางใช้เพื่อบอกให้สองฝ่ายทราบว่าข้อมูลมาจากที่ใดและจะไปที่ใด ส่วนโปรโตคอลนัมเบอร์ใช้บอก ไอพี ว่าจะส่งดาต้าแกรมไปยังชั้นทีซีพี หรือโปรเซสไอพี โดยการทำงานส่วนใหญ่ของไอพีจะเป็นทีซีพีแต่ก็ยังมีโปรโตคอลอื่นๆ ที่ใช้ ไอพีซึ่งในส่วนของ โปรโตคอลนัมเบอร์ จะเป็นการบอกว่าใช้โปรโตคอลอะไร ส่วนสุดท้ายคือผลรวมตรวจสอบส่วนหัวใช้เพื่อตรวจสอบว่าข้อมูลของส่วนหัวไม่เสียหายระหว่างการส่ง หรือไม่ก็ถูกส่งไปยังผิดที่หมายซึ่งมีรูปร่างดังนี้

| Version | IHL | Type of Service | Total Length | |
|---|---|---|---|---|
| Identification | | | Flags | Fragment Offset |
| Time to Live | | Protocol | Header Checksum | |
| Source Address | | | | |
| Destination Address | | | | |
| TCP header, then your data ..... | | | | |

...

*รูปที่ 2.25 โครงสร้างของอินเตอร์เนตโปรโตคอล*

ซึ่งข้อมูลที่ส่งมาจะมีลักษณะดังนี้

IT... IT... IT... IT... IT... IT... IT...

ส่วนแฟลก (flag) และแฟรกเมนต์ออฟเซ็ต (flagment offset) ถูกใช้เพื่อแบ่งดาต้าแกรมเป็นขนาดเล็ก ๆ เนื่องจากข้อมูลที่ส่งไปขนาดใหญ่มากและ เพื่อให้ข้อมูลถูกส่งไปยังการเชื่อมต่อ แต่ละแบบได้ เวลาคงอยู่คือตัวเลขที่ลดลงเรื่อยๆ เมื่อส่งไปยังแต่ละระบบและเมื่อค่านี้เป็น 0 ดาต้าแกรมนี้จะถูกทิ้งไป เพื่อป้องกันการส่งข้อมูลวนในเน็ตเวิร์ค

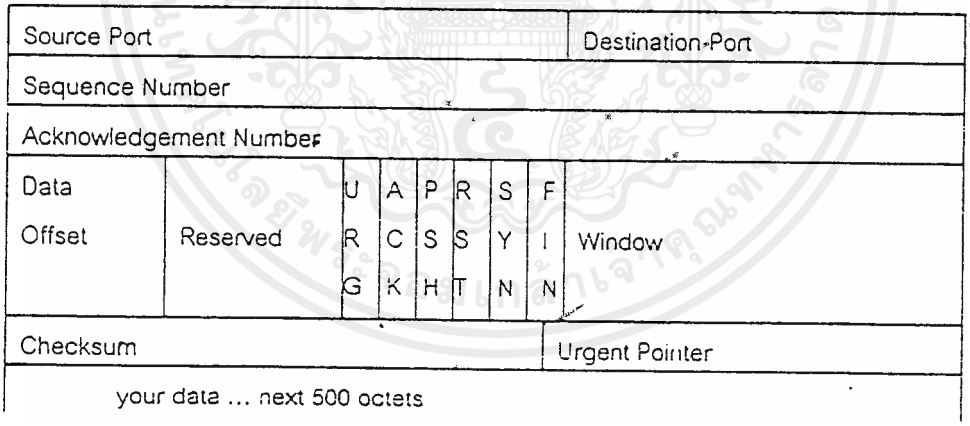
แต่ถ้าในการเชื่อมต่อธรรมดาระหว่างแค่เครื่อง 2 เครื่อง ไม่จำเป็นต้องใช้ส่วนหัวที่มีขนาดใหญ่มากเท่านี้ก็ได้เพื่อลดโอเวอร์เฮด

**2.3.2.3 ระดับชั้นทีซีพี**

ทีซีพี จะแยกข้อมูลเป็นดาต้าแกรมย่อยๆและถูกรวมกลับคืนเมื่อถึงปลายทาง หรือจะถูกส่งอีกครั้งเมื่อข้อมูลสูญหาย ทีซีพีมีการเรียงข้อมูลตามลำดับ และจะส่งข้อมูลต่างๆไปตามเส้นทางของเน็ตเวิร์ค ตัวอย่างเช่น เมื่อใช้ซีเรียลลายน์โมเด็ม (serial line MODEM) จากบ้านมายังโฮสที่ภาควิชา ซึ่งการเชื่อมต่อทางจุดนี้มีแบนด์วิธประมาณ 14.4 กิโลบิตต่อวินาที และเมื่อข้อมูลนี้จะถูกส่งไปยังเราท์เตอร์ (router) ของลาดกระบัง จุดนี้การส่งเป็น 10 เมกกะบิตต่อวินาที และเมื่อข้อมูลถูกส่งยังเกตเวย์ เพื่อไปยังเน็ตเทค (NECTEC) ตรงจุดนี้จะใช้ประมาณ 2 เมกกกะบิตต่อวินาที

ในการทำ มัลติเพิลคอนเนคชั่น(multiple connection) คือการที่สามารถใช้งานได้หลายอย่างในการเชื่อมต่อ ซึ่งทีซีพีจะเป็นตัวกำหนดการทำงานส่วนนี้ งานส่วนนี้เรียกว่า ดีมัลติเพล็กซิ่ง (demultiplexing) ในความเป็นจริงแล้วมีหลายชั้นที่ทำการ ดีมัลติเพล็กซิ่งบน ทีซีพี/ไอพี ซึ่งในส่วนหัวจะมีออกเตตที่บอกถึงโปรเซสที่ทำ

ทีซีพี จะมีส่วนหัวที่เพิ่มเข้าไปในส่วนข้อมูลมีอย่างน้อยกว่า 20 อ็อกเตต ซึ่งจะมีส่วนที่บอกถึงหมายเลขพอร์ต (port number)และหมายเลขลำดับ(sequence)หมายเลขพอร์ต นำไปใช้เพื่อบอกถึงโปรเซสที่จะอ้างอิง สมมุติว่ามี 3 คนกำลังส่งไฟล์ผ่านทีซีพีอาจกำหนดเป็น 1000,1001 และ 1002 สำหรับการส่งครั้งนี้ เมื่อคนแรกส่งดาต้าแกรม ซึ่งจะมาพร้อมกับหมายเลขพอร์ตต้นทาง ไปยังผู้รับอีกฝ่ายก็จะกำหนดหมายเลขพอร์ตสำหรับการติดต่อครั้งนี้ด้วยเช่นกัน และคอมพิวเตอร์เครื่องแรกก็จะทราบด้วยว่าจะส่งไปที่พอร์ตใดในปลายทาง และทุกครั้งที่ส่งข้อมูลกันจะมีหมายเลขลำดับที่ใช้เพื่อระบุถึงลำดับของข้อมูลที่จะส่ง




...

*รูปที่ 2.26 โครงสร้างของทรานสมิสชั่นคอนโทรลโปรโตคอล*

ซึ่งจะมีลักษณะการส่งข้อมูลเป็นดังนี้

T... T... T... T... T... T...

ยังมีหลายส่วนที่ยังไม่ได้อธิบายคือ ส่วนของการจัดการการเชื่อมต่อ ในการที่จะให้การส่งดาต้าแกรมไปยังเป้าหมายได้ ผู้รับจะต้องส่งตอบรับ (acknowlagement) ตอบกลับมา ซึ่งจะมีฟิลด์ของหมายเลขตอบรับ (acknowladgement number) ยกตัวอย่างเช่น ส่งแพ็กเกจโดยมีตอบรับของ 1500 หมายถึงรับข้อมูลอย่างถูกต้องจนถึงอ็อกเตตหมายเลข 1500 ถ้าผู้ส่งยังไม่ได้รับการตอบรับ ในเวลาที่เหมาะสมก็

จะส่งข้อมูลเดิมอีกครั้งหนึ่ง วินโดว์(Window) ถูกนำไปใช้ในการควบคุมจำนวนข้อมูลที่จะส่งไปในแต่ละครั้ง คือจะไม่ตอบรับทุกครั้งที่จะส่งดาต้าแกรมถัดไป ซึ่งจะกำหนดไว้ในส่วนของวินโดว์ เรียกวิธีการนี้ว่า สไลด์ดิ้งวินโดว์(slinding window)

## 2.4 ข้อกำหนดและการโปรแกรมแพ็กเกจไดรเวอร์

## (Packet Driver specification & Programming)

### 2.4.1 ข้อตกลงในเอกสารนี้

ทุกตัวเลขในเอกสารนี้เขียนในรูปแบบภาษาซี โดยเลขฐานสิบเขียนในรูปทั่วคือ 11 เลขฐานสิบหกเขียนอยู่ในรูป 0x0B และเลขฐานแปดเขียนอยู่ในรูป 013 ซึ่งใช้กับทุกๆ ค่าที่อ้างอิงกับเน็ตเวิร์คฮาร์ดแวร์แอดเดรส(ต้นทาง ปลายทาง และมัลติคาสต์) และข้อมูลดีมัลติเพล็กซิ่งสำหรับแพ็กเกจส่วนหัวแพ็กเกจระดับเอ็มเอซี (MAC - level package header) ก่อนที่จะผ่านพ้นไปฟังก์ชั่น send_pkt()

### 2.4.2 ข้อแนะนำและข้อสำคัญ

ในบทนี้จะกล่าวถึงวิธีการเขียนโปรแกรมเชื่อมต่อกับ แพ็กเกจไดรเวอร์ ซึ่งแพ็กเกจไดรเวอร์นั้นต้องง่ายและเป็นพื้นฐานในการเขียนโปรแกรมเชื่อมต่อ ซึ่งสามารถให้หลายโปรแกรม ใช้การ์ดอินเทอร์เฟสที่ดาต้าลิงค์ลีเวล (data link level) ร่วมกันได้ แพ็กเกจไดรเวอร์จะทำการแยกแพ็กเกจที่เข้ามาไปยังโปรแกรมโดยใช้ชนิดแพ็กเกจมาตรฐานของตัวกลาง (Network media's standard packet type) หรือเซอร์วิสแอ็กเซสพอยน์ฟิลด์(Service access point field)

ในรายละเอียดนี้สามารถให้มีการทำโปรโตคอลสแต็ก(protocol stack) ซึ่งเป็นอิสระกับยี่ห้อหรือรุ่นของการ์ดอินเทอร์เฟส ซึ่งแตกต่างไปขึ้นอยู่กับสื่อเน็ตเวิร์ค (network media) เช่นอีเทอร์เนต ริง 802.5, ซีเรียลไลน์ เนื่องจากความแตกต่างในการใช้โปรโตคอลแปลงไปยังฟิสิคัลแอดเดรส(physical address) รูปแบบส่วนหัวและเอ็มทียู (MTU : Maximum Transmission Unit)

แพ็กเกจไดรเวอร์จะจัดเตรียมส่วนเรียกใช้ที่ กำหนดเริ่มต้นใช้งาน(initiate access) สิ้นสุดการใช้งาน (end access) ส่งแพ็กเกจและให้ข้อมูลสถิติบนการ์ดอินเทอร์เฟส และให้ข้อมูลเกี่ยวกับอินเทอร์เฟสในการใช้โปรโตคอล ในแต่ละโปรแกรมร่วมกัน ต้องใช้แพ็กเกจไดรเวอร์ร่วมกัน ผู้ใช้สามารถใช้ทีซีพี/ไอพี เอกซ์เอ็นเอส(XNS) และโปรโตคอลที่ใช้เฉพาะ เช่น เดคเน็ต(DECNET) ไบยัน (bayan's) ไลฟ์เน็ต (lifenet's)และโนเวล(Novell's) โดยไม่ต้องไปเชื่อมยุ่งยากกับการ์ดอินเทอร์เฟส เพียงแต่เรียกไปยังแพ็กเกจไดรเวอร์เท่านั้น

โปรแกรมซึ่งใช้แพ็กเกจไดรเวอร์ สามารถนำไปใช้ในเน็ตเวิร์คอื่นๆ ซึ่งอยู่ในคลาสเดียวกันได้โดยไม่ต้องแก้ไขโปรแกรม เพียงแต่หาแพ็กเกจไดรเวอร์อันใหม่เท่านั้น

ในแพ็กเกจไดรเวอร์ สามารถแบ่งระดับชั้นได้ดังนี้

ฟังก์ชั่นพื้นฐาน(basic function) มีฟังก์ชั่นการทำงานพื้นฐานซึ่งง่ายในการใช้งานเนื่องจากใช้ทรัพยากรน้อย

ฟังก์ชั่นเพิ่มเติม (Extended Function) มีฟังก์ชั่นมากกว่าแบบพื้นฐานสนับสนุนมัลติคาสต์และเก็บสถิติในอินเทอร์เฟส

ฟังก์ชั่นประสิทธิภาพสูง(High-performance Function) สนับสนุนการปรับปรุงประสิทธิภาพ

### 2.4.3 การระบุการ์ดอินเทอร์เฟส

การ์ดอินเทอร์เฟสกำหนดไว้ด้วยตัวเลขสามส่วนสำคัญคือ

**2.4.3.1 คลาส(Class)** ใช้บอกชนิดของสื่อที่อินเทอร์เฟสนี้สนับสนุน เช่น ดีไอเอ็กซ์ (DIX : DEC/Intel/Xerox)อีเทอร์เน็ต, ไออีอีอี 802.3, ไออีอีอี 802.5 (IEEE 802.5), โปรเน็ต 10 (proNET-10) แอปเลทอร์ค(Appletalk), ซีเรียลไลน์,อื่นๆ

**2.4.3.2 ชนิด(type)** ใช้กำหนดรายละเอียดของอินเทอร์เฟสซึ่งสนับสนุนในคลาสนั้นๆ เช่น ในอินเทอร์เน็ตคลาสนั้นระกอบด้วย ชนิด 3COM,3C503,3C505, Interland NI5210 Univation, BICC Data Networks ISOLAN, Ungermann-Bass ฯลฯ ในคลาส ไออีอีอี 802.5 ประกอบด้วยชนิด NIC IBM Token Ring adaptor,Proteon p1340 อื่นๆ

**2.4.3.3 หมายเลข(Number)** ถ้าในเครื่องประกอบด้วย อินเทอร์เฟส มากกว่า หนึ่งคลาส หรือหนึ่งชนิด จะต้องใช้หมายเลขในการระบุถึงความแตกต่าง

คลาสเป็นตัวเลข 8 บิต และชนิดเป็นตัวเลข 16 บิตซึ่งกำหนดโดยเอฟทีพีซอฟแวร์(FTP software) และชนิด 0xFFFF แทนที่ทุกชนิดจะตรงกับทุกอินเทอร์เฟสในคลาส และในหมายเลขจะไม่มีไวล์การ์ด (wildcard) ซึ่ง 0 จะหมายถึงอินเทอร์เฟสอันแรกของคลาสและชนิด

ในข้อกำหนดไม่ได้สนับสนุนมัลติเพิลการ์ดอินเทอร์เฟส (multiple interface card ) ในแพ็กเกจไดรเวอร์เดียวกัน ซึ่งหมายความว่าต้องเรียกแพ็กเกจไดรเวอร์หนึ่งตัวต่อหนึ่งอินเทอร์เฟส ซึ่งต้องเรียกหลายแพ็กเกจไดรเวอร์ และกำหนดอินเทอร์รัป (interrupt) ในแต่ละแพ็กเกจไดรเวอร์ ในกรณีที่ต้องใช้มากกว่าหนึ่งอินเทอร์เฟส ซึ่งโปรแกรมจะต้องตรวจสอบคลาสและชนิดซึ่งได้จากการเรียก driver_info() เพื่อให้มั่นใจว่าใช้แพ็กเกจไดรเวอร์ตรงกับสื่อและรูปแบบแพ็กเกจ

### 2.4.4 การเริ่มทำงานของแพ็กเกจไดรเวอร์

แพ็กเกจไดรเวอร์ถูกอ้างอิง ในโปรแกรมอินเทอร์รัป ในช่วง 0x60 ถึง 0x80 จะอธิบายถึงวิธีการหาอินเทอร์รัป ที่ไดรเวอร์ใช้งานอยู่ ในการกำหนดอินเทอร์รัป จะต้องสามารถเปลี่ยนแปลงได้เพื่อไม่ไปรบกวนกับการทำงานโปรแกรมอื่นๆ

แฮนด์เดลอร์ (handler) สำหรับอินเทอร์รัปกำหนดโดยเริ่มต้นด้วย 3 ไบต์ของโค้ดทำงาน (executable code) อาจจะเป็นคำสั่งกระโดด 3 ไบต์หรือ 2 ไบต์แล้วตามด้วยคำสั่ง เอ็นโอพี (NOP : No Operation) แล้วตามด้วยตัวอักษรที่ไม่มีตัวจบ โดยมีข้อความว่า "PKT DRVR" เพื่อจะหาอินเทอร์รัปที่ถูกใช้โดยแพ็กเกจไดรเวอร์ โปรแกรมจะต้องค้นหาตั้งแต่ 0x60 ถึง 0x80 จนกระทั่งพบข้อความ "PKT DRVR" ใน 12 ไบต์ซึ่งตามหลังจุดเริ่มต้น วิธีการหาแฮดเดอร์ ดูได้จากฟังก์ชั่น access_type()

### 2.4.5 โปรแกรมมิ่งอินเทอร์เฟส (Programming Interfhase)

ทุกฟังก์ชั่นที่ใช้งานโดยซอฟท์แวร์อินเทอร์รัป ทุกครั้งที่มีการเรียกใช้ จะต้องผ่านค่าไปให้ AH รีจิสเตอร์(register) ซึ่งจะเป็นตัวกำหนดฟังก์ชั่นในการใช้งาน

**แฮนด์เดิล(handle)** คือ ค่าตัวเลขซึ่งเกี่ยวเนื่องกับแต่ละชนิดของแม็คลีเวลดีมัลติเพล็กซิ่ง (MAC-level demultiplexing) ซึ่งได้มาจากการเรียกประเภทการใช้งาน (access type) ภายในแพ็กเกจไดรเวอร์จะเป็นพอยน์เตอร์ (pointer)หรือเทเบิ้ลออฟเซต (Table offset)ก็ได้

บางฟังก์ชั่นเรียกใช้ได้กรณีที่เป็น *ฟังก์ชั่นเพิ่มเติม* หรือ *ฟังก์ชั่นประสิทธิภาพสูง* เนื่องจากไม่จำเป็นต่อการใช้งานพื้นฐานของเน็ตเวิร์ค ดังนั้นจึงกำหนดเป็นส่วนที่เพิ่มเติม ซึ่งจะตรวจสอบว่าสามารถใช้งานได้หรือไม่ต้องตรวจสอบที่ driver_info()

เงื่อนไขเริ่มต้น

สามารถใช้แฮนด์เดิลร่วมกันได้เพียง 16 แฮนด์เดิล

การเรียงตัวของบิตข้อมูลที่ส่งในเน็ตเวิร์คตรงกันข้ามกับข้อมูลในการเก็บของคอมพิวเตอร์ทั่วไปยกเว้น 802.5 โทเคนริง นั่นหมายความว่า ค่าประเภทของอีเทอร์ (Ethertype values) ซึ่งส่งไปยัง access_type() จะต้องกลับกัน

#### 2.4.5.1 Driver_info()

| | |
|---|---|
| function | driver_info(handle) |
| input | handle*optional* |
| error return | carry flag set |
| | Error code |
| possible error | BAD_HANDLE |
| non-error return | carry flag clear |
| | Version |
| | Class |
| | Type |
| | Number |
| | Name |
| | Functionality |
| | 1 == basic functions present |
| | 2 == basic and extended present |
| | 5 == basic and high-performance |
| | 6 == basic,high-performance,extended |
| | 255 == not installed |

ในการเรียกฟังก์ชั่นนี้ใช้เพื่อหาข้อมูลเกี่ยวกับ เวอร์ชั่นอินเทอร์เฟส หมายถึง ตัวระบุฮาร์ดแวร์ไดรเวอร์ (Hardware driver) ในสมัยก่อนนั้น แฮนด์เดิลที่ส่งเข้าไปนั้นต้องมีเสมอ แต่ในปัจจุบันเป็นตัวเลือกซึ่งจะมี หรือไม่มีก็ได้ ดังนั้นผู้ใช้ต้องระวัง

**2.4.5.2 access_type()**

| | | |
|---|---|---|
| function | int | access_type(if_class,if_number,type,typelen,reciver) |
| input | | if_class |
| | | if_type |
| | | if_number |
| | | type |
| | | typelen |
| | | receiver() |
| error return | | carry flag set |
| | | error code |
| possible error | | NO_CLASS |
| | | NO_TYPE |
| | | NO_NUMBER |
| | | BAD_TYPE |
| | | NO_SPACE |
| | | TYPE_INUSE |
| non-error | | carry flag clear |
| | | Handle |
| receiver call | | (*receiver)(handle,flag,len[,buffer]) |
| input | | handle |
| | | flag |
| | | len |
| If AX == 1 | | char far *buffer, DS:SI |

เป็นตัวเริ่มต้นในการทำงานของแพ็กเกจไดรเวอร์ ซึ่งมีค่าต่าง ๆ ดังนี้

type ค่าที่ส่งผ่านไปยังฟังก์ชั่นนั้นต้องกำหนดโดย ข้อกำหนดชนิดแพ็กเกจ

typelen คือขนาดเป็นไบต์ของฟิลด์ type ถ้าเป็น 0 หมายความว่าต้องการเรียกทุกแพ็กเกจ

receiver คือพอยน์เตอร์ที่ชี้ไปยังส่วนของโปรแกรมซึ่งถูกเรียกทุกครั้งเมื่อแพ็กเกจได้รับ

เมื่อได้รับแพ็กเกจ ก็จะเรียก receiver สองครั้งด้วยกัน ครั้งแรกเรียกเพื่อขอบัฟเฟอร์(buffer) จากโปรแกรมเพื่อคัดลอกแพ็กเกจลงไป ซึ่งค่า AX จะมีค่าเท่ากับ 0 ซึ่งโปรแกรมจะต้องส่งค่าพอยน์เตอร์ไปยังบัฟเฟอร์ใน ES:DI และถ้าเป็นโปรแกรมไม่มีบัฟเฟอร์จะส่งค่า 0:0 ใน ES:DI และไดรเวอร์จะทิ้งแพ็กเกจนั้นเสีย และจะไม่มีการเรียกครั้งที่สอง

ความยาวของแพ็กเกจเป็นส่วนสำคัญมากโดยเก็บค่าใน CX ค่านี้จะใช้ได้เมื่อค่า AX มีค่าเท่ากับ 0 ซึ่ง receiver สามารถกำหนดบัฟเฟอร์ให้เพียงพอกับขนาดได้ ค่าความยาวนี้รวมถึง ส่วนหัวแม็ค (MAC header) และข้อมูลที่รับได้ แต่ไม่รวมส่วน เอฟซีเอส (FCS : frame check sequence)

ในการเรียกครั้งที่สองนั้น ค่า AX มีค่าเท่ากับ 1 ซึ่งหมายถึง การลอกแพ็กเกจนั้นเสร็จสิ้นแล้ว และโปรแกรมสามารถนำบัฟเฟอร์ไปใช้งานได้ ซึ่งบัฟเฟอร์ที่ใช้งานได้จะส่งมาใน DS:SI

#### 2.4.5.3 release_type()

| | |
|---|---|
| function | release_type(handle) |
| input | handle |
| error return | carry flag set |
| | error code |
| possible error | BAD_HANDLE |
| non-error | carry flag clear |

ฟังก์ชั่นนี้ใช้เพื่อสิ้นสุดการใช้งานโดยเกี่ยวเนื่องกับแฮนด์เดิลที่ได้มาจาก access_Type()

#### 2.4.5.4 send_pkt()

| | |
|---|---|
| function | send_pkt(buffer,length) |
| input | buffer |
| | length |
| error return | carry flag set |
| | error code |
| possible error | CANT_SEND |
| non-error | carry flag clear |

เป็นการเรียกเพื่อส่งข้อมูลขนาด length ไบต์ใน buffer ซึ่งโปรแกรมจะต้องจัดเตรียมทั้งแพ็กเกจรวมทั้งส่วนหัวโลคอลเน็ตเวิร์ค (local network headers) ซึ่งจะไม่ถูกใส่ในไดรเวอร์

#### 2.4.5.5 terminate()

| | |
|---|---|
| function | terminate(handle) |
| input | handle |
| error return | carry flag set |
| | error code |
| possible error | BAD_HANDLE |
| | CANT_TERMINATE |
| non-error | carry flag clear |

เป็นการสิ้นสุดการใช้งานไดรเวอร์ ซึ่งเกี่ยวเนื่องกับแฮนด์เดิล บางทีไดรเวอร์ จะคืนค่าหน่วยความจำให้กับระบบปฏิบัติการเลย ซึ่งจะแตกต่างกับ release_type()

#### 2.4.5.6 get_address()

| | |
|---|---|
| function | get_address(handle,buf,len) |
| input | handle |
| | buf |
| | len |
| error return | carry flag set |
| | error code |
| possible error | BAD_HANDLE |
| | NO_SPACE |
| non-error | carry flag clear |
| | length |

ใช้เพื่อหาค่าฮาร์ดแวร์แอดเดรสของอินเทอร์เฟสการ์ด ใส่ลงไปบน buf ซึ่งมีความยาว len ไบต์ ซึ่งค่าจริง ๆ ของตัวเลขจะส่งคืนใน CX ถ้ามี NO_SPACE error หมายความว่า len นั้นมีขนาดไม่เพียงพอกับความยาวของฮาร์ดแวร์แอดเดรส ถ้าแอด์เดรสถูกเปลี่ยนโดย set_address() แอดเดรสใหม่จะถูกส่งคืนแทน

#### 2.4.5.7 reset_interface()

| | |
|---|---|
| function | reset_interface(handle) |
| input | handle |
| error return | carry flag set |
| | error code |

| | |
|---|---|
| possible error | BAD_HANDLE |
| | CANT_RESET |
| non-error | carry flag clear |

เริ่มอินเทอร์เฟสใหม่อีกครั้ง โดยเกี่ยวเนื่องกับแฮนด์เดิล เพื่อที่จะรู้สถานะ (state) จะทำการหยุดการส่งข้อมูลและกำหนดโหมดการรับ (receiver mode) ใหม่อีกครั้ง ค่าฮาร์ดแวร์แอดเดรสจะเปลี่ยนเป็นค่าเดิม จากหน่วยความจำที่เก็บไว้ ค่ามัลติคาสท์ลิสต์(multicast list) จะถูกลบและโหมดการรับจะเป็น 3 (รับเฉพาะค่าแอดเดรสตัวเองและบรอดคาสท์) ถ้าแฮนด์เดิลเปิดอยู่หลายอัน การกระทำนี้จะไปรบกวนการทำงานของโปรแกรมอื่นที่ใช้อินเทอร์เฟสนี้ร่วมกันได้ ดังนั้น CANT_RESET จะถูกคืนมาแทน

2.4.5.8 get_parameter() *high-performance driver function

| | |
|---|---|
| function | get_parameter() |
| error return | carry flag set |
| | error code |
| possible error | BAD_COMMAND |
| non-error | carry flag clear |
| | struct param |

```
struct param {
        unsigned char   major_rev;      /*revision of แพ็กเกจไดรเวอร์ spec*/
        unsigned char   minor_rev       /*this driver conform to */
        unsigned char   length;         /*length of structure in byte*/
        unsigned char   addr_len/*length of a MAC-layer address*/
        unsigned short  mtu;            /*MTU,including MAC header*/
        unsigned short  multicast_aval  /*buffer size for multicast addr*/
        unsigned short  rcv_bufs;       /*(#of back-to-back MTU rcvs)-1*/
        unsigned short  xmt_bufs        /*(#of successive xmits)-1*/
        unsigned short  int_num;/*Interrupt # to hook for post-EOI
                                        processing,0==none*/
};
```

ความสามารถของโปรแกรม จะได้ประโยชน์จากฟังก์ชั่นนี้ เพื่อที่จะได้ตัวเลขของ ไดรเวอร์พารามิเตอร์(Driver parameter) ซึ่งฟังก์ชันนี้ถูกเพิ่มเติมในเวอร์ชั่น 1.09 และอาจจะไม่ถูกนำไปใช้ในไดรเวอร์อื่นๆ

ค่า major_rev และ minor_rev field คือค่าหมายเลขหลัก และ หมายเลขรองของรุ่นของข้อกำหนด สำหรับข้อกำหนดนี้ ค่า major_rev เป็น 1 และ minor_rev เป็น 9

ค่า length field ใช้เพื่ออธิบายขนาดของ param นี้

ค่า addr_len field เป็นความยาวของแม็คแอดเดรสในขนาดไบต์

ค่า mtu คือ ขนาดใหญ่ที่สุดของแม็คลีเวลแพ็กเกจที่ไดรเวอร์สามารถควบคุมได้ ใน อีเทอร์เน็ต ค่านี้จะถูกกำหนดตายตัว แต่ใน 802.5 และ เอฟดีดีไอ(FDDI) ค่านี้อาจเปลี่ยนแปลงได้

ค่า multicast_aval field คือค่าตัวเลขจำนวนไบต์ซึ่งใช้ในการเก็บ แอดเดรสมัลติคาสท์ซึ่งเป็นฮาร์ดแวร์เมคานิซึม(hardware mechanism) ค่า 0 หมายถึงไม่สนับสนุนมัลติคาสท์

ค่า rcv_bufs และ xmt_bufs ชี้ถึงตัวเลขของแบ็คทูแบ็ครีซีพ (back-to-back receive) หรือการส่ง (transmit) ซึ่งโปรแกรมจะใช้เป็นตัวกำหนด ควบคุมการไหล (flow control) หรือ กลยุทธ์การส่ง (Transmit strategies) ค่า 0 ใน rcv_bufs ใช้โดย ผู้สร้างไดรเวอร์เพื่อที่จะระบุว่าฮาร์ดแวร์นั้นจำกัด เพื่อป้องกันการรับข้อมูลที่ระบบอื่นอาจส่งมาเร็วกว่าก็ได้ ซึ่งแนะนำว่า โปรโตคอลระดับสูงควรจะทำการกำหนดควบคุมการไหลทีละขั้น(lock-step flow control) เพื่อมิให้แพ็กเกจสูญเสีย

ค่า int_num ฟิลด์กำหนดโดย ฮาร์ดแวร์อินเทอร์รัป ซึ่งโปรแกรมสามารถเกาะเพื่อที่จะทำอินเทอร์รัปไทม์โปรโตคอล หลังจาก EOI ซึ่งถูกส่งไปยัง 8259 อินเทอร์รัปคอนโทรลเลอร์และค่า 0 หมายถึงไม่มีอินเทอร์รัป ถ้าโปรแกรมใดเกาะเข้ากับอินเทอร์รัปนี้ และค่าไม่เป็นศูนย์ในเวกเตอร์ จะต้องส่งอินเทอร์รัปลงไปยังลูกโซ่ (chain) และรอพรีดีเซสเซอร์ (predecessor) เพื่อคืนค่าก่อนที่จะทำงานหรือสแต็กสวิทซ์(stack switches)

**2.4.5.9 as_send_pkt() *high-performance driver function**

| | | |
|---|---|---|
| function | int | as_send_pkt(buffer,length,upcall) |
| input | | buffer |
| | | length |
| | | upcall() |
| error return | | carry flag set |
| | | error code |
| possible error | | CANT_SEND |
| | | BAD_COMMAND |
| non-error | | carry flag clear |
| | | buffer available upcall: |
| | | (*upcall)(buffer,result) |
| | | result |
| | | buffer |

แตกต่างจาก send_pkt() ตรงที่ว่า upcall() routine ถูกเรียกเมื่อใช้ข้อมูลโปรแกรมถูกคัดลอกไปจากบัฟเฟอร์แล้ว และโปรแกรมสามารถแก้ไขและ ใช้บัฟเฟอร์ได้โดยปลอดภัย คือไม่มีใครมาเขียนอีก ไดรเวอร์อาจส่งรหัสผิดพลาดที่ไม่เป็น 0 (non-zero error code) ไปยัง upcall() ถ้าการคัดลอกนั้นผิดพลาด หรือข้อผิดพลาดที่เกิดขึ้น ในกรณีอื่นๆหมายถึงสำเร็จ แม้ว่าแพ็กเกจจะยังไม่ถูกส่งไปจริงๆ สังเกตว่าบัฟเฟอร์ที่ส่งไปยัง send_pkt() ถูกกำหนดว่าสามารถแก้ไขได้เมื่อเรียกเสร็จ ในขณะที่ as_send_pkt() บัฟเฟอร์จะถูกจัดลำดับโดยไดรเวอร์ถ้ามีข้อผิดพลาดเกิดขึ้น upcall จะไม่ทำงาน

**2.4.5.10 set_rcv_mode() *extended driver function**

| | |
|---|---|
| function | set_rcv_mode(handle,mode) |
| input | handle |
| | mode |
| error return | carry flag set |
| | error code |
| possible error | BAD_HANDLE |
| | BAD_MODE |
| non-error | carry flag clear |

เป็นการกำหนดการทำงานของการปรับข้อมูลแพ็กเกจโดยเกี่ยวเนื่องกับแฮนด์เดิลที่ต้องการจะรับ ซึ่งมีโหมดต่างๆ ดังนี้

1 ไม่รับแพ็กเกจ
2 รับเฉพาะที่ส่งมาอินเทอร์เฟสนี้
3 โหมด 2 รวมกับบรอดคาสท์แพ็กเกจ
4 โหมด 3 รวมกับลิมิตเตดมัลติคาสท์แพ็กเกจ
5 โหมด 3 รวมกับมัลติคาสท์แพ็กเกจ
6 ทุกแพ็กเกจ

<u>หมายเหตุ</u>

อินเทอร์เฟสไม่ทุกอันที่สนับสนุนการรับแบบ ทุกแพ็กเกจ และรีซีฟเวอร์โหมดมีผลกระทบต่ออินเทอร์เฟสโดยตรง ไม่เกี่ยวกับแฮนด์เดิล

โหมด 3 เป็นค่าเริ่มแรกและถ้าฟังก์ชั่น set_rcv_mode() ไม่ได้กำหนดไว้จะถือเป็นโหมด 3

**2.4.5.11 get_rcv_mode() extended driver function**

| | |
|---|---|
| function | get_rcv_mode(handle,mode) |
| input | handle |

| | |
|---|---|
| error return | carry flag set |
| | error code |
| possible error | BAD_HANDLE |
| non-error | carry flag clear |
| | mode |

ส่งค่าโหมดการรับปัจจุบัน

**2.4.5.12 get_statistics() extended driver function**

| | |
|---|---|
| function | get_statistics(handle,mode) |
| input | handle |
| error return | carry flag set |
| | error code |
| possible error | BAD_HANDLE |
| non-error | carry flag clear |
| | stats |

```
struct statistics {
        unsigned long   packets_in;         /* Totals across all handles */
        unsigned long   packets_out
        unsigned long   bytes_in;           /* including MAC headers */
        unsigned long   bytes_out
        unsigned long   errors_in           /* Totals across all error types */
        unsigned long   errors_out
        unsigned long   packets_lost;       /* No buffer from receiver(), card */
                                            /* out of resources, etc. */
}
```

ส่งค่าพอยน์เตอร์ไปยังโครงสร้างข้อมูลสถิติสำหรับอินเทอร์เฟสซึ่งค่าเก็บอยู่ในรูป 80xx 32บิต

**2.4.5.13 set_address() *extend driver function**

| | |
|---|---|
| function | set_address(addr,len) |
| input | addr |
| | len |
| error return | carry flag set |
| | error code |

| | |
|---|---|
| possible errors | CANT_SET |
| | BAD_ADDRESS |
| non-error | carry flag clear |
| | length |

การเรียกนี้ใช้เมื่อโปรแกรม หรือโปรโตคอลสแตก ต้องการเจาะจงใช้ฮาร์ดแวร์แอดเดรสหมายเลขนั้น ๆ

BAD_ADDRESS หมายความว่า ค่า len น้อยไปหรือมากไป หรือ แอดเดรสไม่ถูกต้อง แพ็กเกจไดรเวอร์ จะปฏิเสธการเปลี่ยนแอดเดรส ถ้ามีแฮนด์เดิลมากกว่าหนึ่งเปิดใช้งานอยู่

**2.4.6 หมายเลขฟังก์ชั่นเรียกและพารามิเตอร์**

ตัวเลขต่อไปนี้ ถูกกำหนดเพื่อใช้ในการทำงานของ แพ็กเกจไดรเวอร์ ซึ่งกำหนดในค่า รีจิสเตอร์ AH เพื่อเรียกใช้งาน แพ็กเกจไดรเวอร์

| | |
|---|---|
| driver_info | 1 |
| access_type | 2 |
| release_type | 3 |
| send_pkt | 4 |
| terminate | 5 |
| get_address | 6 |
| reset_interface | 7 |
| +get_parameter | 10 |
| +as_send_pkt | 11 |
| *set_rcv_mode | 20 |
| *get_rcv_mode | 21 |
| *set_multicast_list | 22 |
| *get_multicast_list | 23 |
| *set_statistics | 24 |
| *get_statistics | 25 |

\+ หมายถึง ฟังก์ชั่นประสิทธิภาพสูง

* หมายถึงฟังก์ชั่นเพิ่มเติม

ค่า AH ตั้งแต่ 128 ถึง 255 สงวนไว้ใช้สำหรับการพัฒนาอื่นๆ ซึ่งนอกเหนือจากข้อกำหนด

**รหัสผิดพลาด(Error codes)**

ในการเรียก แพ็กเกจไดรเวอร์ ถ้ามีข้อผิดพลาดเกิดขึ้น จะมีการเซตค่าแฟลกตัวทด (carry flag) และค่ารหัสผิดพลาดจะถูกกำหนดในรีจิสเตอร์ DH (ซึ่งรีจิสเตอร์นี้จะไม่ถูกใช้ในการส่งค่าผ่าน ฟังก์ชั่นแต่ถูกใช้เพื่อส่งรหัสผิดพลาดกลับมา ) ซึ่งกำหนดดังนี้

| | | |
|---|---|---|
| 1 | BAD_HANDLE | หมายเลขแฮนด์เดิลผิด |
| 2 | NO_CLASS | ไม่พบคลาสที่กำหนด |
| 3 | NO_TYPE | ไม่พบชนิดที่กำหนด |
| 4 | NO_NUMBER | ไม่พบหมายเลขที่กำหนด |
| 5 | BAD_TYPE | กำหนดชนิดแพ็กเกจผิด |
| 6 | NO_MULTICAST | อินเทอร์เฟสนี้ไม่สนับสนุนมัลติคาสท์ |
| 7 | CANT_TERMINATE | ไม่สามารถสิ้นสุดการทำงานแพ็กเกจไดรเวอร์ |
| 8 | BAD_MODE | กำหนดวิธีการแพ็กเกจผิด |
| 9 | NO_SPACE | การทำงานผิดพลาดไม่มีที่ว่าง |
| 10 | TYPE_INUSE | ชนิดที่กำหนดไม่สามารถใช้งานได้เนื่องจากถูกใช้งานอยู่ |
| 11 | BAD_COMMAND | คำสั่งนอกเหนือจากที่กำหนด |
| 12 | CANT_SEND | ไม่สามารถส่งแพ็กเกจได้เนื่องจากฮาร์ดแวร์ |
| 13 | CANT_SET | ไม่สามารถเปลี่ยนฮาร์ดแวร์แอดเดรสได้เนื่องจากมีแฮนด์เดิลมากกว่า 1 เปิดอยู่ |
| 14 | BAD_ADDRESS | รูปแบบ หรือ ขนาดของฮาร์ดแวร์แอดเดรสผิด |
| 15 | CANT_RESET | ไม่สามารถเริ่มใหม่ได้เนื่องจากมีแฮนด์เดิลมากกว่า 1 เปิดอยู่ |

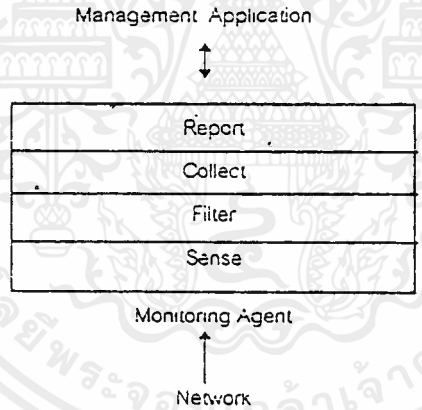
# บทที่ 3
# หลักการของเน็ตเวิร์คมอนิเตอร์ริง

## 3.1 หลักการของเน็ตเวิร์คมอนิเตอริง
## ( Design of Network Monitors )

### 3.1.1 ฟังก์ชั่นทั่วไปของเน็ตเวิร์คมอนิเตอร์ริงเอเจนท์

มอนิเตอร์ริงเอเจนท์ (monitoring agent) บางตัวไม่ได้มีฟังก์ชั่นครบทั้งหมด เช่น เรียลไทม์ดาต้าอานาไลเซอร์ (real-time data analyzer) จะแสดงผลข้อมูลไปยังหน้าจอแสดงผล โดยตรงไม่ต้องเก็บลงในสื่อบรรจุ (storage media) เช่น ฮาร์ดดิสก์ แต่จะมีฟังก์ชั่นพื้นฐานเหมือนกันดังนี้

1. ส่วนรับข้อมูล (Sensing)
2. ส่วนกรองข้อมูล (Filter)
3. ส่วนรวบรวมข้อมูล (Collecting)
4. ส่วนแสดงผล (Reporting)




รูปที่ 3.1 หลักการของแต่ละฟังก์ชั่นในเน็ตเวิร์คมอนิเตอริง

### 3.1.2 ประเภทของการมอนิเตอร์

เน็ตเวิร์คมอนิเตอร์ริ่งเอเจนท์ สามารถแบ่งออกเป็นประเภทได้ 2 วิธีด้วยกัน คือ

1. แบ่งตามเลเยอร์ที่เอเจนท์ทำการมอนิเตอร์อยู่
2. แบ่งตามลักษณะการมอนิเตอร์คือ เอเจนท์นั้นเป็นแบบ อินทิเกรตเต็ด (Integrated)หรือเอ็กเทอร์นอล (Eternal) กับ ส่วนประกอบหรือทรัพยากรที่ทำการมอนิเตอร์

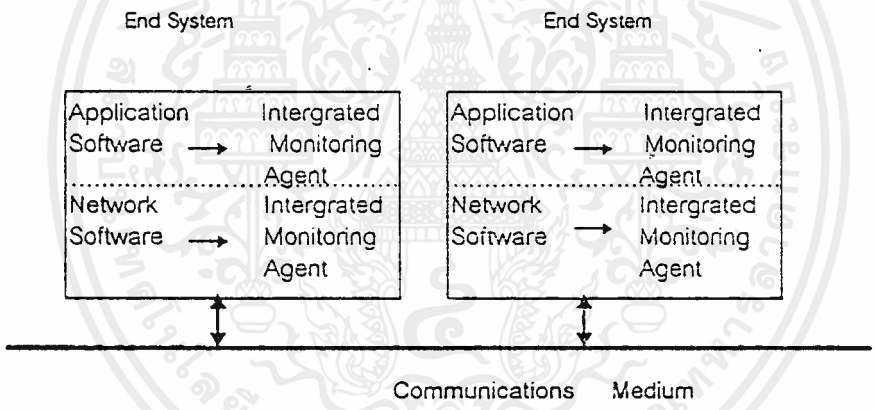
วิธีการแบ่งประเภททั้ง 2 วิธี มีผลต่อการออกแบบและการใช้งานมอนิเตอร์ ดังนั้นเราจะมากล่าวถึงแต่ละประเภทว่ามีลักษณะอย่างไร

**3.1.2.1 อินทิเกรตเด็ดมอนิเตอร์เอเจนท์ ( Integrated monitoring agents)**

อินทิเกรตเด็ดมอนิเตอร์เอเจนท์ สามารถรวมไว้เป็นส่วนหนึ่งของเน็ตเวิร์คแอพพลิเคชั่น หรือ ในตัวของเน็ตเวิร์คซอฟแวร์เลยก็ได้ ซึ่งในกรณีหลังนั้น เราต้องทำการรวมเอาส่วนต่าง ๆ ต่อไปนี้เข้ามาไว้ด้วย

- เน็ตเวิร์คเซอร์วิส (network service) ของบางระดับชั้น เช่น รีโมทโพรซีเดอร์คอล (RPC : Remote Procedure Call ) คอมมิวนิเคชั่นเซอร์วิส ( communication service )
- เน็ตเวิร์คซอฟแวร์ ในพื้นฐานระบบปฏิบัติการ เช่น เน็ตเวิร์คดีไวซ์ ไดรเวอร์ ( network device drivers)
- ส่วนหนึ่งของ ส่วนประกอบเน็ตเวิร์ค ( network component ) เช่น ซอฟแวร์ หรือ เฟิร์มแวร์ (firmware) ที่ทำงานในฟรอนท์เอ็น คอนเซนเตรเตอร์ (front-end concentrator)

จากตัวอย่างที่กล่าวมานี้ เราจำเป็นต้องมี ตัวนับเหตุการณ์ ( event counters ) ในทุก ๆ เน็ตเวิร์คเลเยอร์ ซอฟแวร์ที่ทำหน้าที่นี้ ก็คือ มอนิเตอร์เอเจนท์นั่นเอง




รูปที่ 3.2 ไดอะแกรมของเน็ตเวิร์คมอนิเตอร์เอเจนท์

**ประโยชน์ของอินทิเกรตเด็ดมอนิเตอร์**

1. สามารถทำการวิเคราะห์ในระยะไกล (remote diagnostic capability) ได้
2. ในกรณีที่มอนิเตอร์ริงเอเจนท์ รวมอยู่กับแอพพลิเคชั่น เฉพาะอย่าง มันสามารถสนองความต้องการของแอพพลิเคชั่นนั้น ๆ มากกว่าจะทำฟังก์ชั่นทั่วไป
3. ประหยัด เนื่องจากบางระบบมีบริการของมอนิเตอร์อยู่แล้ว ไม่ต้องไปเสียค่าใช้จ่ายในส่วนนี้อีก

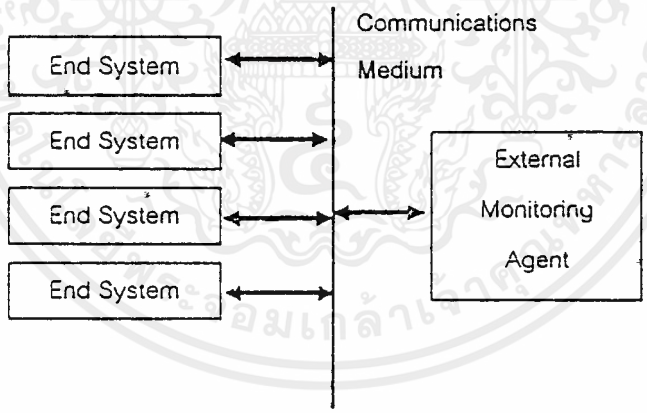
**ข้อเสียของอินทิเกรตเต็ดมอนิเตอร์**

1. ถ้ามอนิเตอร์ริงเอเจนท์ทำงานบนรูปแบบ (platform) เดียวกันกับตัวบริการด้าน เครือข่าย และแอพพลิเคชั่น ตัวมอนิเตอร์จะทำให้ประสิทธิภาพของ แอพพลิเคชั่นลดลง
2. ยุ่งยากในการเก็บข้อมูลที่ได้จากมอนิเตอร์ริงเอเจนท์ และค่าหน่วงเวลาที่เกิดขึ้นในช่วงเก็บข้อมูล จะทำให้ข้อมูลบางตัวไม่สามารถเก็บได้ทัน
3. มอนิเตอร์ริงเอเจนท์ ต้องปรับเปลี่ยนไปตามซอฟแวร์ที่มันไปรวมอยู่ด้วย
4. ตัวมอนิเตอร์ จะให้รูปแบบของรายงานที่ต่างกัน สามารถแก้ได้โดยใช้รูปแบบของรายงานที่เป็นมาตรฐาน
5. การมอนิเตอร์เครือข่าย มีความสำคัญน้อย เมื่อเทียบกับแอพพลิเคชั่น หรือ โอเอส

**3.1.2.2 เอกซ์เทอร์นอลมอนิเตอร์ (External monitoring agents)**

เอกซ์เทอร์นอลมอนิเตอร์เอเจ้นท์ทำงานในคอมโพเนนท์ที่ไม่เกี่ยวข้องกับซอฟแวร์ที่ทำหน้าที่เน็ตเวิร์คเซอร์วิส เอเจนท์ชนิดนี้สามารถดักจับได้เฉพาะ แชร์สเตท(share state) ที่ถูกอ้างจากเน็ตเวิร์คโปรโตคอลเท่านั้น และสามารถดัดแปลงได้ง่าย นิยมใช้ในระบบแลน

เอกซ์ทอร์นอลมอนิเตอร์เอเจนท์ไม่ต้องติดต่อกับระบบแลนสามารถนำไปติดตั้งในระหว่างพอยน์ทูพอยน์คอมมิวนิเคชั่นลิงค์(point to point communication link) หรือติดต่อกับบัสในระบบคอมพิวเตอร์




รูปที่ 3.3 ไดอะแกรมของเอกซ์เทอร์นอลมอนิเตอร์

**ประโยชน์ของเอกซ์เทอร์นอลมอนิเตอร์เอเจนท์**

1. สามารถทำเป็นฮาร์ดแวร์ได้โดยไม่ไปแย่งทรัพยากรกับเน็ตเวิร์คเซอร์วิส และแอพพลิเคชั่น ช่วยลดสาเหตุที่ทำให้ประสิทธิภาพของเน็ตเวิร์คลดลงโดยมอนิเตอร์ได้ สามารถใช้ฮาร์ดแวร์และซอฟท์แวร์พิเศษ เพื่อใช้สำหรับนำมาทำมอนิเตอร์ริงซึ่งจะไม่มีในมอนิเตอร์ริงเอเจนท์แบบอินทิเกรตเต็ด
2. ประหยัด
3. เนื่องจากความเป็นอิสระในการใช้งาน ทำให้ปรับปรุงและขยายการใช้งานได้

**ข้อเสีย**

1. ความปลอดภัยในการจัดการน้อย
2. ข้อมูลที่ได้ไม่สามารถรับประกันได้ว่าแสดงสถานะจริง ๆ ของคอมโพเนนท์ที่เรามอนิเตอร์จะเป็นเพียงค่าประมาณเท่านั้น
3. เนื่องจากต้องเก็บข้อมูลของเอ็นด์ซิสเต็มจำนวนมาก ตัวเอเจนท์ต้องมีความสามารถสูงมาก ทำให้เพิ่มความยุ่งยากในการผลิตเพื่อทำให้ไม่มีข้อจำกัดในการทำงาน

### 3.1.3 การประยุกต์ใช้งาน

ปัจจัยที่มีผลต่อการตัดสินใจใช้อินทิเกรตเตดหรือเอกซ์เทอร์นอลมอนิเตอร์ริงเอเจนท์มีดังนี้

1. แบนด์วิธ (Bandwidth) ของเน็ตเวิร์คคอมมิวนิเคชั่นมีเดีย (network communication media)
2. สิทธิในการใช้สื่อนั้น ๆ
3. ความเร็วของหน่วยประมวลผล
4. ราคาของหน่วยความจำ

**เน็ตเวิร์คแบนด์วิธ(Network bandwidth)**

ถ้าแบนด์วิธของเน็ตเวิร์คเพิ่มขึ้น จะมีผลต่อเอกซ์เทอร์นอลมอนิเตอร์ริงเอเจนท์เพราะว่ามีข้อมูลจำนวนมากขึ้นที่ต้องการประมวลผล

**ใช้สื่อร่วมกัน(Shared medium)**

ใช้เน็ตเวิร์คที่เป็นแบบพอยน์ทูพอยน์ลิงค์ที่เหมาะสมที่จะใช้เอ็กซ์เทอร์นอลมอนิเตอร์ริงเอเจนท์ เนื่องจาก เอเจนท์หนึ่งๆสามารถมอนิเตอร์การติดต่อระหว่าง 2 ระบบปลายทาง(End system) เท่านั้น การจะให้มีการมอนิเตอร์ทุกๆส่วนในเน็ตเวิร์คต้องใช้ค่าใช้จ่ายสูง

จากเหตุผลที่กล่าวมา ทำให้เราใช้เอกซ์เทอร์นอลมอนิเตอร์ริงเอเจนท์ สำหรับทดสอบ (diagnostic) เป็นส่วนใหญ่ โดยใช้ในการวินิจฉัยเฉพาะลงไปในกรณีที่มีปัญหาเกิดขึ้นในส่วนใดส่วนหนึ่ง

### 3.1.4 สิ่งที่จะมอนิเตอร์ในแต่ละเลเยอร์

ในแต่ละเน็ตเวิร์คเลเยอร์ต้องการรายละเอียดในการมอนิเตอร์ต่างกันดังนี้

1. ดาต้าลิงค์เลเยอร์ ใช้มอนิเตอร์จริงในการตรวจสอบซอฟท์แวร์และฮาร์ดแวร์ที่ผิดพลาด ซึ่งเป็นผลมาจากการคอรัปชั่น(corruption)หรือการสูญหายของข้อมูล

2. เน็ตเวิร์คเลเยอร์ ใช้มอนิเตอร์ริงในการรายงานถึงเส้นทางการเชื่อมต่อ(วงจร) ว่าใช้การได้หรือใช้การไม่ได้

3. ทรานสปอร์ตเลเยอร์ ในเลเยอร์นี้มอนิเตอร์ริงสามารถช่วยในการจัดการเน็ตเวิร์คและการทำการวางแผน โดยดูจากระดับที่สัมพันธ์ของแต่ละทรานสปอร์ตโปรโตคอล

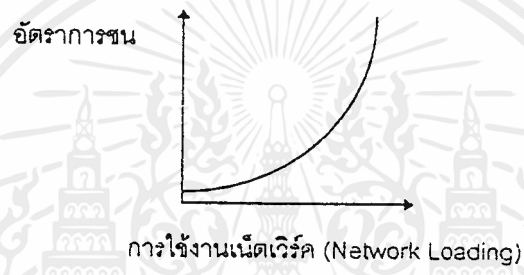
4. เซสชั่นเลเยอร์ ดูเกี่ยวกับปริมาณการใช้งาน(work load) ของเน็ตเวิร์คที่ใช้โดยแอพพลิเคชั่น และ/หรือ ผู้ใช้ซึ่งเป็นประโยชน์ต่อการทำ โหลดบาลานซิ่ง (load balancing)และการทำแอกเคานท์ติ้ง แอพพลิเคชั่น (accouting application)

5. แอพพลิเคชั่นเลเยอร์ ใช้ตรวจสอบความผิดพลาดของ เซิฟเวอร์โปรเซส(server process) ในไค์ลน์/เซิฟเวอร์แอพพลิเคชั่นเป็นต้น

## 3.2 ประสิทธิภาพและปัญหาของเน็ตแวร์แลน

## (Netware LANs Performance and Troubleshooting)

### 3.2.1 บทบาทของการสื่อสารบนเน็ตเวิร์ค




รูปที่ 3.4 ประสิทธิภาพของอีเทอร์เนต

ในการวัดปริมาณการใช้งานนั้น เราวัดขนาดของข้อมูลต่อเวลาเทียบกับแบนด์วิธ(bandwidth) ซึ่งถ้ามีการใช้งานน้อย นั่นคือน้อยกว่า 5 เปอร์เซนต์ของแบนด์วิธทั้งหมด (total bandwidth) เมื่อมีการใช้งานเพิ่มขึ้นอัตราการชนกัน (collision rate) ของข้อมูลยิ่งมีมากขึ้น โดยสถิติกล่าวว่าที่ 30 เปอร์เซ็นต์ ประสิทธิภาพการใช้งานของเน็ตเวิร์คจะลดลงอย่างรวดเร็ว

เพราะฉะนั้นสิ่งที่เราต้องวัดเพื่อที่จะตรวจสอบการใช้งานเน็ตเวิร์คคือ

- การใช้งานของเน็ตเวิร์ค
- การใช้งานสูงสุด(peak)
- การใช้งานโดยเฉลี่ย

ซึ่งที่กล่าวมานี้จะช่วยในการนำไป รีอาร์เรนท์จิ้ง (rearranging) งานที่ใช้ทรัพยากรของเน็ตเวิร์คสูงได้

ข้อแนะนำ ควรจะวัดทั้งสัปดาห์ โดยเว้นระยะห่างในการจับเป็นครึ่งชั่วโมง

### 3.2.2 สิ่งที่ควรจะวัดในเน็ตเวิร์คทั่วไป

#### 3.2.2.1 เปอร์เซนต์การใช้งานของเน็ตเวิร์ค

ทำได้โดยการวัดทราฟฟิก (traffic) บนเน็ตเวิร์คในช่วงเวลาสั้นๆ สำหรับเปอร์เซนต์การใช้งานของอีเทอร์เนตที่ดีต้องไม่เกิน 15 เปอร์เซ็นต์ นอกเหนือจากนี้ ก็จะมีการวัดการชนกัน (collision) ซึ่งมีผลกระทบต่อประสิทธิภาพ

#### 3.2.2.2เน็ตเวิร์คทรูพุท(Network throughput)

วัดจำนวนไบต์ทั้งหมดที่ส่งผ่านเน็ตเวิร์คในเวลาใดๆซึ่งจะวัดได้ 2 แบบคือ

1.ข้อมูลดิบ(raw data) จำนวนไบต์ทั้งหมดที่ได้รับ

2.ข้อมูลที่ใช้จริง(usable data)จำนวนยูเซอร์ดาต้าในแต่ละโปรโตคอล

เนื่องจากโปรโตคอลต่างๆมีโอเวอร์เฮด (overhead) ซึ่งมีผลต่อดาต้าทรูพุท (data throughput) ด้วยเหตุนี้เราจึงนำมาใช้พิจารณาในการออกแบบเน็ตเวิร์คด้วย

ข้อสังเกต อีเทอร์เนต 10 เม็กกะบิตต่อวินาที(Megabit per Second) ไม่ใช่หมายถึงให้ดาต้าทรูพุทสูงสุด 10 เม็กกะบิตต่อวินาที ในการออกแบบเราคำนึงว่า 2.5 เม็กกะบิตต่อวินาที คือทรูพุทสูงสุด (Maximum throughtput).

#### 3.2.2.3 เวลาตอบสนองของโปรเซสไฟล์เซิฟเวอร์(Response time of the file server process)

คือการวัดเวลาตอบสนอง (response) ของไฟล์เซิฟเวอร์ (file sever) ต่อการร้องขอ(request) เป็นการวัดประสิทธิภาพของระบบปฏิบัติการและไฟล์เซิฟเวอร์ที่ทำงานอยู่

#### 3.2.2.4 เน็ตเวิร์คคอนเวอร์เซชั่น(Network conversation)

อินดิเซ้นต์ (indecence) และตำแหน่ง (location) ของคอนเวอร์เซชั่น เป็นสิ่งสำคัญในการนำมาพิจารณาถ้าเกิดมีคอขวด (bottlenecks) ขึ้นที่เน็ตเวิร์คฮาร์ดแวร์ เช่น บริดจ์ หรือ เราท์เตอร์

#### 3.2.2.5 เก็บบันทึกข้อผิดพลาดในเน็ตเวิร์ค(recording network error)

เก็บบันทึกข้อผิดพลาดที่เกิดขึ้นในเน็ตเวิร์ค เพื่อที่สามารถดูข้อผิดพลาดที่เกิดขึ้นภายหลังได้

### 3.2.3 รูปแบบของแพ็กเกจในอีเทอร์เนตและประสิทธิภาพ

#### 3.2.3.1 อีเทอร์เนตเฟรม(ethernet frame)

| Size in octets | 7 | 1 | 6 | 6 | 1 | 0-1500 | 0-46 | 4 |
|---|---|---|---|---|---|---|---|---|
| Preamble | SFD | Desunation address | Source address | DLF | Data | PAD | Checksum |

SFD,Start of frame delimiter

DLF,Length of data field

PAD,(optional) padding field

รูปที่ 3.5 โครงสร้างแพ็กเกจอีเทอร์เนต

**3.2.3.2 อีเทอร์เนตแอดเดรส (Ethernet adress)**

3.2.3.2.1 บรอดคาสท์ (Broadcast)

3.2.3.2.2 มัลติคาสท์ (Multicast)

**3.2.3.3 ส่วนหัวของโปรโตคอลระดับบน(Higher level protocol header)**

3.2.3.3.1 เน็ตเวิร์คเลเยอร์

ส่วนหัวโนเวลไอพีเอ็กซ์(Novell's IPX header)

ส่วนหัวอินเทอร์เนตโปรโตคอล(IP header)ขนาดเล็กสุดมีขนาด 20 ไบต์

3.2.3.3.2 ทรานสปอร์ตเลเยอร์

ส่วนหัวโนเวลเอสพีเอ็กซ์(Novel's SPX header) มีขนาด 12 ไบต์

ทีซีพีเฮดเดอร์ ขนาดเล็กที่สุดมีขนาด 20 ไบต์

เพราะฉะนั้นถ้าเราใช้ ทีซีพี/ไอพี โปรโตคอลจะมีโอเวอร์เฮดอย่างน้อย 66 ไบต์ส่วนหัวโนเวลไอพีเอ็กซ์/เอสพีเอ็กซ์ จะมีโอเวอร์เฮด 68 ไบต์

**3.2.3.4 ผลกระทบของขนาดแพ็กเกจกับประสิทธิภาพ**

ขนาดของแพ็กเกจยิ่งมีขนาดเล็กเท่าไร โอเวอร์เฮดที่เกิดจากส่วนหัวของโปรโตคอล(protocol header) ก็จะยิ่งมากขึ้นเท่านั้น เช่นเมื่อมีการส่ง ตัวอักษรเพียงตัวเดียวจาก เทอร์มินอล (terminal) จะพบว่าข้อมูลที่พบจริงจะเป็น 2 เปอร์เซนต์ของเฟรมเท่านั้น แต่ถ้าส่งข้อมูลที่มีขนาดแพ็กเกจสูงสุด พบว่าข้อมูลผู้ใช้คิดเป็น 95 เปอร์เซนต์ของแพ็กเกจ จะเห็นว่าขนาดแพ็กเกจเป็นปัจจัยสำคัญของประสิทธิภาพของเน็ตเวิร์ค

ในการส่งข้อมูล ถ้าใช้เฟรมขนาดใหญ่จะช่วยลดจำนวนของแพ็กเกจลง ซึ่งเท่ากับช่วยลดจำนวนการเกิด การชน ดังนั้น โอเวอร์เฮดที่เกิดจาก การชนและการส่งใหม่อีกครั้งจึงลดไปด้วย แต่การเพิ่มขนาดของเฟรมให้ใหญ่ขึ้นก็จะทำให้เกิดผลเสียเช่นกัน ดังนี้ ดูเหมือนเฟรมขนาดใหญ่จะทำให้จำนวนของที่ว่างบนเน็ตเวิร์คลดลงในขณะที่ทราฟฟิกมีการใช้งานสูง- และขนาดบัฟเฟอร์ที่ต้องการโดยซอฟท์แวร์ที่ควบคุมเน็ตเวิร์คอินเทอร์เฟสต้องเพิ่มขึ้น

แต่จากการค้นคว้าพบว่าในทางปฏิบัติ ข้อสำคัญที่ใช้พิจารณาคือ ดาต้าทรูพุท(ตรงข้ามกับมินิมั่มดีเลย์ (minimum delay) ที่ต้องการในระบบเรียลไทม์ (real-time system) ดังนั้นขนาดเฟรมควรจะมีขนาดใหญ่

**3.2.4 สิ่งที่ต้องทำการวัดประสิทธิภาพ**

**3.2.4.1 สิ่งที่ต้องทำการวัด สำหรับวัดประสิทธิภาพของเน็ตเวิร์ค**

- การใช้งานเน็ตเวิร์ค(protocol in use)
- การกระจายขนาดของเฟรม(frame size distribution)
- โหนดที่ใช้งานสูงสุด(busy nodes)

- โหนดที่ไม่ได้ใช้งาน(idle node)
- โหนดที่ไม่ได้ตอบสนอง(unresponsive node)
- การสนทนาสูงสุด(busiest conversation)
- เวลาและระดับการใช้งานสูงสุด(peak load time and levels)
- เวลาและระดับการใช้งานต่ำสุด(minimum load time and levels)
- แบนด์วิธที่ใช้ในแต่ละโหนด(bandwidth usage by node)
- แบนด์วิธที่ใช้ในแต่ละโปรโตคอล(bandwidth usage by protocol)
- ทั้งหมดนี้เป็น ข้อกำหนดขั้นต่ำของเครื่องมือในการวัดที่ควรพิจารณา นอกเหนือจากนี้แล้ว ยังมีอย่างอื่นที่ต้องการวัด ซึ่งมีประโยชน์เมื่อใช้ในการออกแบบ และการพัฒนาสำหรับประสิทธิภาพของเน็ตเวิร์คคือ
- โปรโตคอลที่ใช้ในการสนทนา(protocol use in conservation)
- กำหนดค่าในการแจ้งเตือนเหตุร้ายที่เกิดขึ้น
- การเตือนเมื่อมีเหตุเกิดขึ้นแก่เน็ตเวิร์ค
- สิ่งสำคัญอย่างอื่นที่ช่วยในการวัดประสิทธิภาพคือ
- สร้างทราฟฟิกเน็ตเวิร์ค(generating network traffics) ช่วยให้สามารถจำลองเน็ตเวิร์คได้โดยการสร้างแพ็กเกจและส่งเข้าไปในเน็ตเวิร์ค
- ผลของการแสดงต้องช่วยให้เข้าใจง่าย และสามารถจับเก็บ(Capture) ข้อมูลที่กำหนดเพื่อนำมาวิเคราะห์ภายหลังได้

**3.2.4.2 การวัดประสิทธิภาพ**

**3.2.4.2.1 การใช้งานแบนด์วิธ(bandwidth usage)**

หาค่าเฉลี่ยการใช้งานช่วงเวลาคงที่ (ทุก ๆ 1 นาที หรือทุกๆ 1 วินาที) ทำได้โดยนับจำนวนทราฟฟิกในช่วงเวลาที่กำหนด หารด้วยค่าทรูพุทสูงสุด ค่าที่คำนวณได้จะไม่ต่ำกว่า 5 เปอร์เซนต์และพิจารณาดังนี้

- 10-15 เปอร์เซนต์ แสดงว่าเน็ตเวิร์คมีการใช้งานน้อยถึงปานกลาง
- 25 เปอร์เซนต์ แสดงว่าเน็ตเวิร์คมีการใช้งานสูงสุด เน็ตเวิร์คแจม์มิ่ง(network jamming) การตอบสนองช้า มีแพ็กเกจเสียมาก หรือ มีการส่งซ้ำปริมาณมาก

**3.2.4.2.2 ข้อผิดพลาดในการส่ง(Transmission error)**

แพ็กเกจที่เกิดการชนกันจะต้องทำการส่งใหม่อีกครั้งหนึ่งโดยส่วนใหญ่จะถูกรายงานว่าซีอาร์ซีผิดพลาด หรือแพ็กเกจสั้นขนาดผิดปกติ(runts) ฯลฯ ซึ่งสาเหตุมาจากการชนกัน สำหรับเน็ตเวิร์คที่มีการใช้งานสูง(ประมาณ 20 เปอร์เซนต์) อัตราการชนกันที่ยอมรับได้จะอยู่ระหว่าง 1-2 เปอร์เซนต์

ถ้าทั้งการใช้งานแบบแบนด์วิธและ อัตราแพ็กเกจเสีย(failed packet rates)สูงทั้งคู่ หมายความว่าประสิทธิภาพของเน็ตเวิร์คไม่ได้ดีดังที่ควรเป็นซีอาร์ซีผิดพลาดอาจมีสาเหตุมาจากเน็ตเวิร์คอินเทอร์เฟส

เช่น เราทราบว่าแต่ละโหนดมีการใช้งานต่ำ ในขณะที่เน็ตเวิร์คมีการใช้งานสูง สาเหตุหลักส่วนใหญ่จะเกิดจากเน็ตเวิร์คอินเทอร์เฟส สาเหตุอย่างอื่นเช่น ตัวเชื่อมต่อ(connector) หลวม

**3.2.4.2.3 การวัดที่จำเป็นสำหรับการออกกแบบเพื่อแบ่งเน็ตเวิร์ค (Measurements needed for dessign to partition network)**

ทำได้หลายวิธีดังนี้

- แบ่งตามชนิดของโปรโตคอลที่ใช้ออกจากกันใช้ ในกรณีที่พีซีแลนด์ (PC LANs)และไมโครคอมพิวเตอร์อยู่ในเน็ตเวิร์คเดียวกัน เป็นวิธีที่ง่ายต่อการนำไปฏิบัติ แต่ในกรณีที่มีการใช้งานที่หลากหลายอยู่ด้วยกัน ไม่เหมาะอย่างยิ่งที่จะแบ่งเป็นโปรโตคอล
- แบ่งตามฟิสิคอลแอดเดรส ใช้ในกรณีที่เรามี เวิร์คกรุป(workgroup) ในแต่ละตำแหน่งที่ต้องการติดต่อกับเน็ตเวิร์คเป็นบางครั้ง ส่วนใหญ่จะติดต่อกันเองภายในกลุ่มการที่เราจะทราบว่าผู้ใช้ใดจะอยู่ในกลุ่มเดียวกัน เราจะต้องเก็บข้อมูลของการสนทนาในแต่ละโหนด

**3.2.4.2.4 ขนาดของแพ็กเกจและประสิทธิภาพ (Packet size and performance)**

ทรูพุทของดาต้าที่ใช้งานจริง ( usable data throughput ) กับ เน็ตเวิร์คทรูพุทแตกต่างกัน แอพพลิเคชั่นจะเป็นตัวกำหนดขนาดแพ็กเกจที่เหมาะสม(optimum packet size) เช่นถ้าแอพพลิเคชั่นมีการรับส่งข้อมูลที่ขนาดใหญ่ ซึ่งจำเป็นต้องแบ่งข้อมูลออกเป็นหลายแพ็กเกจ เรากำหนดให้ใช้ขนาดแพ็กเกจสูงสุดของ เน็ตเวิร์คโปรโตคอลที่ใช้จะดีที่สุด ( 1518 ไบต์ในกรณีอีเทอร์เนต ) ซึ่งจะช่วยลดจำนวนของแพ็กเกจที่ใช้ในการส่งข้อมูลและช่วยลดผลของโปรโตคอลโอเวอร์เฮดด้วย

ปัจจัยที่เป็นข้อจำกัดของ การกำหนดขนาดของเฟรมคือ โครงสร้างของเน็ตเวิร์คซึ่งการใช้บริดจ์ และ เราท์เตอร์ จะไม่ยอมให้แพ็กเกจใหญ่ผ่าน(บางที) หรือว่าแอพพลิเคชั่นอื่นต้องการพารามิเตอร์ที่ต่างกันออกไป

ไม่มีกฎตายตัวสำหรับการกำหนดขนาดของเฟรม ทางเดียวก็คือ เมื่อมีการเปลี่ยนแปลงเราต้องคอยดูผลที่เกิดขึ้น ซึ่งได้จากข้อมูลการมอนิเตอร์ริง(monitoring information) โดยไม่เพียงแต่ดูเฉพาะส่วนที่มีการเปลี่ยนแปลง เราต้องดูผลที่เกิดกับเน็ตเวิร์คโดยรวมด้วย

**3.2.4.2.5 โหนดใช้งานและไม่ใช้งาน(Active and inactive nodes)**

โหนดที่ไม่สนองตอบ คือ โหนดที่ได้รับการติดต่อแต่ไม่ตอบสนองต่อการร้องขอ แต่ไม่ได้ส่งการติดต่อ

**3.2.4.2.6 กำหนดช่วงเวลาใช้งานสูงสุด(definition of peak usage time)**

เราต้องบันทึกเวลาที่เกิด การใช้งานสูงสุด และการใช้งานน้อยสุด เพื่อนำไปทำแผน การทำงาน เช่น ออโตเมติคแบ็คอัพ (automatic backup) การอัพเดตครั้งใหญ่( large batch update) หรือการส่งข้อมูล

มอนิเตอร์ริงแพ็กเกจ(monitoring package) จะช่วยได้มาก และยังสามารถทราบทราฟฟิกเพื่อจำลองการทำงานได้

### 3.3 ภาษาที่ใช้ในการพัฒนาซอฟแว่ร์

เนื่องจากการใช้งานจะต้องติดต่อกับ การ์ดเชื่อมต่อเน็ตเวิร์ค(Network interface card)โดยตรง และเพื่อให้มีประสิทธิภาพมากขึ้นโดยไม่ผ่านขั้นตอนการทำงานของสแต็กโปรโตคอล (Stack protocol) อื่นๆ และประกอบกับได้รับตัวโปรแกรมมาจากการพัฒนาต่อเนื่องมาจากรุ่นที่แล้ว จึงเลือกภาษาปาสคาล (Pascal) ซึ่งง่ายในการพัฒนาเครื่องมือให้ใช้อย่างง่าย

ภาษาปาสคาลเหมาะสำหรับการพัฒนาโปรแกรม บนพื้นฐานระบบดอส(DOS) และเนื่องจากเคยใช้งานมาก่อนจึงง่ายต่อการเข้าใจ และการพัฒนาและมีเครื่องมือช่วยในการทำส่วนติดต่อผู้ใช้

การพัฒนาโปรแกรมนั้นจะต้องเรียกแพ็กเกจไดรเวอร์ โดยเรียกผ่าน อินเทอร์เฟสอินเทอร์รัปเวคเตอร์(Interface interrupt vector) ของแพ็กเกจไดรเวอร์ และเพื่อให้โปรแกรมนั้นง่ายต่อการพัฒนาจึงได้แบ่งเป็นโมดูลย่อยๆ แล้วจึงเรียกผ่านอินเทอร์เฟสอินเทอร์รัปเวคเตอร์

# บทที่ 4

# การวางแผนออกแบบและการสร้าง

## 4.1 หลักการเบื้องต้นในการสร้าง Software

### 4.1.1 ศึกษาระบบ (System Study)

ทำการศึกษาระบบโดยพิจารณาถึงความต้องการ และศึกษาสิ่งแวดล้อมที่มีความสัมพันธ์กับระบบที่กำลังศึกษาอยู่ เช่น การใช้งานในระบบเครือข่ายจำเป็นต้องใช้อะไรบ้างในการวิเคราะห์

### 4.1.2 ศึกษาถึงความเป็นไปได้ (Feasibility Study)

ศึกษาความเป็นไปได้ของระบบว่าเราสามารถนำข้อมูลอะไรได้บ้าง ใช้วิเคราะห์อะไรได้บ้าง

### 4.1.3 วิเคราะห์ความต้องการ (Requirement Analysis)

ศึกษาและวิเคราะห์ระบบปัจจุบันที่มีอยู่ พิจารณาข้อดีและข้อเสียของระบบ ทำความเข้าใจระบบให้เป็นอย่างดี และหาความต้องการ (Requirement) ของระบบ อาจจะทำการจำลองระบบต่าง ๆ เพื่อช่วยในการทำความเข้าใจ

### 4.1.4 กำหนดความต้องการ (Requirement Definition)

เป็นกิจกรรมที่ทำการเปลี่ยนข้อมูลต่าง ๆ ที่รวบรวมได้จากการวิเคราะห์ให้เป็นเอกสารที่กำหนดความต้องการต่าง ๆ ที่ตรงตามที่ผู้ใช้ต้องการ เอกสารต้องเขียนด้วยภาษาที่เข้าใจง่าย ซึ่งผู้ใช้ระดับต้นสามารถอ่านเข้าใจได้ง่าย

### 4.1.5 กำหนดรายละเอียดความต้องการ (Requirement Specification)

เป็นรายละเอียด และข้อกำหนดของแต่ละความต้องการ อธิบายถึงหน้าที่ความต้องการ ในรายละเอียดที่ลึกลงไปควรเขียนด้วยภาษาที่ไม่มีความกำกวม เพื่อไม่ให้สับสนระหว่างผู้อ่านและผู้เขียน

### 4.1.6 ออกแบบซอฟท์แวร์ (Software Design)

ออกแบบซอฟท์แวร์โดยเลือกภาษาที่ใช้ในการพัฒนา ออกแบบวิธีการที่ใช้ในการเขียนซอฟท์แวร์เลือกใช้กลยุทธ์ในการเขียนซอฟท์แวร์ เช่น การเขียนออบเจ็ค-โอเรียนเต็ด (Object-Oriented) เพิ่มรายละเอียดลงไปในข้อกำหนดที่ทำขึ้นในการออกแบบ โครงสร้างข้อมูลที่จะใช้ การออกแบบแบ่งเป็น

- ออกแบบสถาปัตยกรรม (Architectural Design)
- ข้อกำหนดเบื้องต้น (Abstract Specification)
- ออกแบบส่วนติดต่อ (Interface Design)
- ออกแบบส่วนเชื่อมต่อ (Component Design)
- ออกแบบโครงสร้างข้อมูล (Data Structure Design)
- ออกแบบวิธีการทำงาน (Algorithm Design)

### 4.1.7 จัดทำและทดสอบการใช้งาน (Implementation and Unit Testing)

จัดทำแยกทดสอบโมดูลย่อยที่สร้างขึ้นว่า ทำงานถูกต้องตามต้องการหรือไม่

### 4.1.8 รวบรวมและทดสอบระบบ (Integration and System Testing)

เมื่อทำการทดสอบโมดูลย่อย ๆ เสร็จแล้ว ก็นำโมดูลต่าง ๆ มารวมกันเป็นโปรแกรมที่ต้องการ แล้วทำการทดสอบการทำงานโดยรวมอีกครั้งหนึ่ง

### 4.1.9 ใช้งานและดูแล (Operation and Maintenance)

นำซอฟท์แวร์นั้นไปใช้งานจริง และบางครั้งอาจมีการเปลี่ยนแปลงเกิดขึ้นจึงต้องมีการดูแล ดังนั้นซอฟท์แวร์ที่ดีควรออกแบบให้สามารถดูแลได้ง่าย

## 4.2 การวางแผนและพัฒนาระบบ

ขั้นตอนในการวางแผนและพัฒนาระบบมีดังต่อไปนี้

- ศึกษาการทำงานของเนื้อหาข้อมูลในส่วนต่าง ๆ ที่ใช้ในการออกแบบโปรแกรมโดยพิจารณาถึงข้อดีข้อเสียของแต่ละส่วนว่าควรจะปรับปรุงอะไร โดยศึกษาถึงการทำงานของโปรแกรมเดิม
- ทำการศึกษาลักษณะการทำงานของโปรแกรมเก่าทั้งหมด และทำการออกแบบโครงสร้างของโปรแกรมใหม่ เพื่อให้เพิ่มประสิทธิภาพในส่วนต่าง ๆ ที่จำเป็นในการใช้งาน
- ออกแบบขั้นตอนหลักของโปรแกรมดังนี้
  - โครงสร้างข้อมูลของแต่ละโมดูล
  - อัลกอริทึมโมดูลย่อยต่าง ๆ
  - ส่วนแสดงผล
- เลือกถึงวิธีการทำงานของโปรแกรมให้เหมาะสมตามส่วนต่าง ๆ ที่กำหนดไว้คือเลือกพัฒนาโปรแกรมบนภาษาปาสคาลและใช้เครื่องมือต่าง ๆ

## 4.3 โครงสร้างข้อมูลของแต่ละโมดูล

### 4.3.1 โมดูลแสดงคอนเวอร์เซชั่น (Display Conversation)

- ที่อยู่ผู้ส่ง
- ที่อยู่ผู้รับ
- จำนวนแพ็กเกจจากผู้ส่งไปยังผู้รับ
- จำนวนแพ็กเกจจากผู้รับไปยังผู้ส่ง
- จำนวนไบต์จากผู้ส่งไปยังผู้รับ
- จำนวนไบต์จากผู้รับไปยังผู้ส่ง
- จำนวนแพ็กเกจจากผู้ส่งไปยังผู้รับต่อวินาที
- จำนวนแพ็กเกจจากผู้รับไปยังผู้ส่งต่อวินาที
- จำนวนไบต์จากผู้ส่งไปยังผู้รับต่อวินาที
- จำนวนไบต์จากผู้รับไปยังผู้ส่งต่อวินาที

- เวลาที่เริ่มใช้งาน
- เวลาสุดท้ายที่ใช้งาน
- โปรโตคอลที่ใช้งาน

การเก็บข้อมูลลงไฟล์ประกอบด้วย 2 ส่วนด้วยกันคือ ส่วนหัว และ ส่วนข้อมูล ในส่วนหัวนั้นจะเก็บข้อมูลที่จำเป็นโดยบอกถึงสถิติข้อมูลโดยรวม ดังต่อไปนี้

- เวลาเริ่มต้นเก็บข้อมูล
- เวลาสิ้นสุดเก็บข้อมูล
- ช่วงเวลาที่เก็บข้อมูล
- หมายเลขที่อยู่ของเครื่องที่เก็บ
- จำนวนแพ็กเกจทั้งหมด
- จำนวนไบต์ทั้งหมด
- จำนวนข้อผิดพลาดทั้งหมด
- จำนวนการสูญเสียทั้งหมด

ส่วนของข้อมูลนั้นจะเก็บข้อมูลที่จับได้ ดังนี้

- ส่วนหัวของแพ็กเกจ
- ส่วนของข้อมูล
- ความยาวของข้อมูล
- เวลาที่ได้รับแพ็กเกจ

**4.3.2 โมดูลแสดงโปรโตคอล (Display Protocol)**

ส่วนนับจำนวนโปรโตคอล

- นับจำนวนแพ็กเกจ
- จำนวนไบต์

การเก็บจำนวนของโปรโตคอลในแต่ละเลเยอร์

- เลเยอร์ 2
- เลเยอร์ 3
- เลเยอร์ 4 มีการแยกเก็บตามชนิดของโปรโตคอลในเลเยอร์ 3 และ 2 ดังนี้
    - IPX Ethernet II
    - IPX Ethernet 802.3
    - IPX Ethernet SNAP
    - IPX Ethernet 802.2
    - IP Ethernet II
    - IP Ethernet SNAP

- เลเยอร์สูงขึ้นไป มีการแยกเก็บตามชนิดของโปรโตคอลในเลเยอร์ 4 และ 2 ดังนี้
  - TCP Ethernet II
  - TCP Ethernet SNAP
  - UDP Ethernet II
  - UDP Ethernet SNAP

### 4.3.3 โมดูลเก็บสถิติ

โครงสร้างของข้อมูลเป็นดังนี้

- วันที่ ,เวลา (Date,Time)
- แพ็กเกจต่อวินาที (Packet per second)
- การใช้งาน (Utilize)
- อัตราการเกิดข้อผิดพลาด (Error rates)
- อัตราการสูญเสียข้อมูล (Drop per second)
- กิโลไบต์ต่อวินาที (KiloBytes per second)
- บรอดคาสท์ต่อวินาที (Broadcast per second)
- อันเดอร์ไซส์ต่อวินาที (Undersize per second)
- โอเวอร์ไซส์ต่อวินาที (Oversize per second)

## 4.4 ส่วนอัลกอริทึมของโปรแกรม

ช่วงการออกแบบวิธีการทำงานในส่วนต่าง ๆ ซึ่งได้แบ่งเป็น โมดูลหลักแยกจากกันเพื่อง่ายในการพัฒนาโปรแกรมโดยมีส่วนต่าง ๆ ดังนี้

- การจับข้อมูล
- การกรองแพ็กเกจ
- การแจ้งเตือนภัย
- การวิเคราะห์และแสดงผล

### 4.4.1 การจับข้อมูล

ส่วนเริ่มต้นของการเก็บแพ็กเกจ โดยจะต้องคำนึงถึงเรื่องการรับแพ็กเกจ ซึ่งมักจะมีปัญหาตรงที่ไม่สามารถรับแพ็กเกจได้ทัน ทำให้ต้องสูญเสียแพ็กเกจไป ดังนั้นควรจะมีบัฟเฟอร์เพื่อสำรองข้อมูลและทำการจัดลำดับ เพื่อให้สามารถได้ค่าที่ใกล้เคียงความจริงมากที่สุด

การใช้งานจะเรียกผ่านแพ็กเกจไดรเวอร์ ซึ่งต้องติดต่อผ่านทางบริการอินเทอร์รัป (User Interface Interrupt Service) ที่มีให้ ซึ่งจะผ่านข้อมูลมายังโมดูลต่าง ๆ ที่เรากำหนด แล้วค่อยรวบรวมข้อมูลหรือจัดทำสถิติต่อไป

### 4.4.2 การกรองแพ็กเกจ

เมื่อผ่านการกรองขั้นแรกมาแล้วก็จะทำการเก็บข้อมูลลงในบัฟเฟอร์ ในที่นี้สามารถเลือกที่จะเก็บข้อมูลในบัฟเฟอร์ที่เป็นหน่วยความจำ (Memory) หรือฮาร์ดดิสก์ (Harddisk) ก็ได้

ถ้าไม่ได้กำหนดการกรองจะถือว่าให้เก็บทุกแพ็กเกจ ซึ่งอาจทำให้บัฟเฟอร์มีขนาดใหญ่และอาจมีข้อมูลที่เราไม่สนใจรวมอยู่ด้วย ในกรณีนี้จะเป็นการยากที่จะดูแพ็กเกจที่สำคัญหรือมีปัญหา เนื่องจากมีข้อมูลมาก ดังนั้นโปรแกรมสามารถจะทำการกรองอีกขั้นหนึ่งเพื่อให้สามารถดูได้เฉพาะแพ็กเกจที่ต้องการ

#### 4.4.2.1 การกรองขั้นแรก

แบ่งเป็น 2 ชนิดด้วยกันคือ ลักษณะของแพ็กเกจ หรือ ค่าของฟิลด์

##### 4.4.2.1.1 ลักษณะของแพ็กเกจ

ในโปรแกรมสามารถที่จะกรองขั้นแรกได้ซึ่งลักษณะขึ้นอยู่กับรูปแบบแพ็กเกจและ ความถูกต้อง ยกตัวอย่างเช่น

- ทุกแพ็กเกจ
- ทุกแพ็กเกจที่ดี
- ทุกแพ็กเกจที่ผิดพลาด
- แพ็กเกจที่มีขนาดผิดพลาด
- แพ็กเกจที่มีขนาดตามต้องการ

ซึ่งเกณฑ์ดังกล่าวนี้ เป็นอิสระกับชนิดของ โปรโตคอลที่จับได้ เช่น ในเครือข่าย อีเทอร์เนต แพ็กเกจที่ดี มีขนาดตามที่ต้องการจะเป็น โปรโตคอลทีซีพี/ไอพี หรือเน็ตแวร์ก็ได้ ดังตาราง แสดงถึงคุณสมบัติแพ็กเกจและความหมาย

| ลักษณะแพ็กเกจ | ความหมาย |
|---|---|
| ทุกแพ็กเกจ | ทุกแพ็กเกจทั้งดีและผิดพลาด |
| ทุกแพ็กเกจที่ดี | ทุกแพ็กเกจที่ไม่มีข้อผิดพลาด |
| แพ็กเกจที่มีขนาดผิดพลาด | แพ็กเกจที่มีขนาดน้อยกว่า 64 หรือมากกว่า 1518 ไบต์ |
| แพ็กเกจที่มีขนาดตามต้องการ | แพ็กเกจที่มีขนาดตรงกับที่กำหนดไว้ |

รูปที่ 4.1 ตารางแสดงประเภทของการรับแพ็กเกจ

ในการกรองสามารถเลือก ขนาดแพ็กเกจที่ต้องการได้ โดยกำหนดขนาดแพ็กเกจต่ำสุด และสูงสุด ดังตัวอย่างต่อไปนี้ ต้องการเก็บข้อมูลซึ่งสั้นกว่าปกติ (แพ็กเกจที่มีขนาดน้อยกว่า 64 ไบต์และ CRC ถูกต้อง) อาจจะเลือกโดยกำหนดดังนี้คือ เลือกขนาดแพ็กเกจที่มีขนาดน้อยกว่า 64

##### 4.4.2.1.2 ค่าของฟิลด์

ในการกรองแพ็กเกจโดยการกำหนดประเภทส่วนหัว ประเภทของโปรโตคอลระดับเน็ตเวิร์ค โปรโตคอลระดับขนส่ง และโปรโตคอลระดับบนที่ต้องการนั้น จะต้องบอกถึงตำแหน่งของข้อมูลของโปรโตคอลที่นำมาใช้ในการกรอง เช่น ต้องการเก็บข้อมูลที่เป็น RIP บนแลนเน็ตแวร์ ขั้นแรกก็ต้องระบุ

ว่าใน ฟิลด์ซ็อกเกตของผู้ส่ง (Source Socket field) มีค่า 0x0453 และในฟิลด์ที่อยู่ของผู้รับ (Destination Address field) มีค่า FF-FF-FF-FF-FF-FF

#### 4.4.2.2 การกรองขั้นสุดท้าย

เมื่อคุณใช้การกรองขั้นสุดท้าย (หรือเรียกอีกอย่างว่าการกรองก่อนการแสดงผล) ก่อนที่จะใช้ต้องพิจารณาดูว่า ข้อมูลที่อยู่ในบัฟเฟอร์นั้น มีข้อมูลที่เราต้องการหรือไม่ ดังเช่น คุณต้องการเก็บทุกแพ็กเกจของเน็ตแวร์ที่ใช้เฟรมแบบ อีเทอร์เนต 802.2 ในบัฟเฟอร์ ซึ่งผลที่ได้อาจจะมากเกินไปจึงต้องทำการกรองเฉพาะแพ็กเกจที่ต้องการ โดยกรองขั้นสุดท้ายเพื่อเอาเฉพาะแพ็กเกจที่เป็นเน็ตแวร์แซฟ (SAP) เป็นต้น

### 4.4.3 การแจ้งเตือนภัย

เป็นส่วนที่สำคัญช่วยให้สามารถดูแลระบบได้ง่ายขึ้นซึ่งจะเตือนเราให้ทราบถึงเหตุการณ์ต่าง ๆ ที่เกิดขึ้นแทนที่เราจะเฝ้าดูเหตุการณ์ตลอดเวลา ซึ่งจะเตือนเมื่อมีเหตุการณ์ไม่ปกติ ดังเช่น การเพิ่ม ของข้อผิดพลาดที่เกิดขึ้นในเครือข่าย เพื่อให้สอดคล้องกับการใช้งานในแต่ละเครือข่ายจำเป็นต้องมี การกำหนดเทรโชลด์ (Threshold) ให้ตรงกับลักษณะของเครือข่ายนั้น ๆ

การเตือนโดยเทรโชลด์นั้นอยู่บนพื้นฐานของสถิติที่เก็บดังเช่น การใช้งาน แพ็กเกจต่อวินาที ข้อผิดพลาด ในขณะที่การคำนวณสถิตินั้นก็จะเปรียบเทียบกับค่าเทรโชลด์ว่าจะมีการเตือนหรือไม่

<u>ขั้นแรก</u> เป็นขั้นตอนแรกในการวิเคราะห์คือการรวบรวมเก็บสถิติทั้งหมด สถิติทั้งหมดรวมถึงข้อมูลเกี่ยวกับแพ็กเกจที่พบในเครือข่าย ซึ่งแตกต่างจากแพ็กเกจที่ได้จากการกรอง

<u>ขั้นสอง</u> คือการกรองชนิดของแพ็กเกจที่จะเก็บในบัฟเฟอร์ (Buffer) เพื่อใช้ในการตรวจสอบหรือคำนวณสถิติ นับว่าเป็นการกรองขั้นแรก

<u>ขั้นสาม</u> คือการเก็บข้อมูลลงในบัฟเฟอร์ ซึ่งอาจทำได้ 2 วิธีด้วยกัน คือให้เขียนทับข้อมูลที่เก่าที่สุดโดยข้อมูลที่ใหม่ที่สุด (FIFO) หรือหยุดการเก็บข้อมูล

<u>ขั้นสุดท้าย</u> คือการกรองในขั้นแสดงผลซึ่งจะขอกล่าวในช่วงถัดไป

การทำสถิติทั้งหมดนั้น จะต้องคำนวณถึง

- เปอร์เซนต์การใช้งาน
- กิโลไบต์ต่อวินาที
- ข้อผิดพลาดต่อวินาที
- แพ็กเกจต่อวินาที
- การกระจายของขนาดแพ็กเกจ

ดังรูปเมื่อมีแพ็กเกจขนาด 64 ไบต์ เป็นโปรโตคอลเน็ตแวร์บรอดคาสท์ผ่านก็จะถูกเก็บในบัฟเฟอร์ที่จะวิเคราะห์ ก่อนที่จะถูกเก็บนั้นก็จะคำนวณสถิติทั้งหมดและการกรองแพ็กเกจขั้นแรกเสียก่อน ในช่วง

ของการเก็บสถิติทั้งหมด จะเก็บถึงการใช้งาน กิโลไบต์ต่อวินาที แพ็กเกจต่อวินาที และการกระจายของขนาดแพ็กเกจ

เมื่อได้รับแพ็กเกจที่ผิดพลาด การ์ดจะแสดงถึงข้อผิดพลาด โปรแกรมก็จะเพิ่มตัวนับจำนวนข้อผิดพลาดต่อวินาทีและ ทำการแก้ไขเพิ่มเติมสถิติโดยรวมซึ่งเก็บข้อผิดพลาดต่อไปนี้

- ความยาวผิดพลาด (Length errors)
- ข้อผิดพลาดจากซีอาร์ซีหรือการจัดเรียง (CRC/Alignment errors)

เมื่อได้รับแพ็กเกจขนาดมากกว่า 1518 ไบต์ ค่าโอเวอร์แพ็กเกจก็จะเพิ่มขึ้นรวมถึงค่าผิดพลาดทั้งหมด จำนวนของแพ็กเกจและข้อผิดพลาดเรื่องขนาด

การเตือนโดยเทรโชลด์สามารถตั้งได้โดยตั้งค่าให้เป็น 1 โปรแกรมก็จะเตือนเมื่อมีข้อมูลที่มีความยาวยาวกว่าปกติ (ยาวกว่า 1518 ไบต์ )

### 4.4.4 ส่วนออกแบบการแสดงผล

ในการรายงานผลนั้นสามารถรวบรวมข้อมูลสำคัญต่าง ๆ เกี่ยวกับเครือข่าย เช่น สถิติของการใช้งานของเราท์เตอร์ สถิติการใช้งานของไดรเวอร์ และอื่น ๆ อย่างไรก็ตามเราได้ยึดหลักลักษณะสำคัญที่ควรจะมีในการแสดงผลของประสิทธิภาพระบบเครือข่าย ซึ่งส่วนสำคัญนั้นคือ

- อัตราการใช้งาน (Utilization)
- อัตราข้อผิดพลาด (Error rate)
- จำนวนแพ็กเกจต่อวินาที (Packet per second)
- จำนวนกิโลไบต์ต่อวินาที (Kilobytes per second)
- โฮสที่กำลังใช้งานอยู่มากที่สุด (Most active servers)
- จำนวนแพ็กเกจทั้งหมด (Total Packets)
- จำนวนไบท์ทั้งหมด (Total Bytes)

#### 4.4.4.1 การแสดงผลแนวโน้มของการใช้งาน (Utilization Trends)

การที่ค่าการใช้งานสูงขึ้นเป็นลักษณะโดยปกติของเครือข่ายซึ่งมีการเจริญเติบโต ตามกาลเวลา ในขณะที่มีเวิร์คสเตชั่น (Work Station) ใหม่และมีโปรแกรมเพิ่มขึ้นในเครือข่าย มีผู้ใช้และข้อมูลเพิ่มขึ้นตามสภาพการส่งข้อมูลของสายระบบ (Cabling System) ซึ่งแนวโน้มของการใช้งานจะสามารถตระเตรียมและตัดสินใจถึงความต้องการของการขยายตัวในระบบเครือข่ายได้เป็นอย่างดี

การแสดงเป็นแผนภูมิถึงการใช้งานโดยใช้กราฟจะเป็นการใช้งานที่ง่ายขึ้น โดยที่แผนภูมิแนวโน้มการใช้งานแสดงข้อมูลเป็นช่วงเวลา ดังเช่น วัน หรือ สัปดาห์ หรือ เดือน ซึ่งกราฟจะสูงขึ้นหรือลดลงตามปริมาณการใช้งานของเครือข่ายเช่น ในตอนเช้าเมื่อทุกคนเข้ามาใช้ระบบเครือข่าย หรือตอนสำรอง การใช้งานระบบช่วงนี้ ระบบเครือข่ายจะมีการใช้งานมาก ซึ่งจะทำให้กราฟสูงขึ้น ซึ่งโครงสร้างในการเก็บข้อมูลแบบกราฟนั้นอาศัยข้อมูลพื้นฐานดังต่อไปนี้

- วันที่ (Date)
- เวลา (Time)
- จำนวนแพ็กเกจต่อวินาที (Packets per second)
- จำนวนกิโลไบต์ต่อวินาที (KiloBytes per second)
- ค่าเฉลี่ยของความผิดพลาด (Average Errors)
- การใช้งาน (Utilization)

**4.4.4.2 อัตราการเกิดข้อผิดพลาด (Error Rate)**

จะมีการแสดงข้อมูลแนวโน้มของการเกิดข้อผิดพลาดขึ้นในเครือข่าย ซึ่งทำให้แน่ใจว่าเราทราบถึงชนิดของข้อผิดพลาดที่เกิดขึ้นในระบบเครือข่ายทุกช่วงเวลา ในขณะที่เครือข่ายเจริญเติบโตขึ้น อัตราการเกิดข้อผิดพลาดย่อมเพิ่มขึ้นด้วยเช่นกัน เปรียบเสมือนอาการของระบบเครือข่าย ไม่ว่าจะเป็นการเกิดคอขวดที่สายหรือการต่อสายผิดพลาดหรือการที่ส่วนประกอบเกิดเสีย

เมื่อระบบเครือข่ายใช้งานมากการเกิดข้อผิดพลาดยิ่งมากตามไปด้วย แต่อย่างไรก็ตามถ้าพบว่ามีการผิดพลาดเพิ่มขึ้นแต่การใช้งานกลับไม่เพิ่ม อาจเป็นไปได้ที่ส่วนประกอบของเครือข่าย อาจมีปัญหา เช่น อาจเกิดข้อผิดพลาดที่ตัวการ์ดเชื่อมต่อเครือข่ายหรือตัวรับ

**4.4.4.3 แพ็กเกจต่อวินาที**

ตัวเลขที่แสดงค่าแพ็กเกจต่อวินาทีบนเครือข่าย จะช่วยให้ทราบถึงปริมาณการจราจรบนสายซึ่งไม่เหมือนกันกับค่าที่ใช้งาน เนื่องจากค่าการใช้งานนั้นคำนวณมาจากจำนวนกิโลไบต์บนเครือข่ายต่อวินาที ซึ่งแพ็กเกจมีขนาดไม่แน่นอนและไม่สัมพันธ์กับการใช้งาน การใช้งานอาจจะเพิ่มขึ้นเมื่อมีจำนวนแพ็กเกจเพิ่มขึ้นหรือขนาดของแพ็กเกจเพิ่มขึ้น

ในแลนเน็ตแวร์ (Netware LAN) ค่าของแพ็กเกจต่อวินาทีที่บ่งบอกถึงการร้องขอตอบกลับแพ็กเกจและข้อมูลซึ่งมีการใช้บริการอยู่ในเครือข่าย ถ้าค่าจำนวนแพ็กเกจต่อวินาทีเพิ่ม แต่ค่าการใช้งานไม่เพิ่ม นั่นอาจหมายความว่ามีการเพิ่มจำนวนของแพ็กเกจขนาดเล็ก ๆ ซึ่งทำให้การใช้งานไม่เพิ่มขึ้น

**4.4.4.4 กิโลไบต์ต่อวินาที**

โดยแนวทางของกิโลไบต์ต่อวินาที เราสามารถจะคาดคะเนค่าจริงของการไหลของข้อมูลในเครือข่าย ซึ่งการคำนวณสถิติการใช้งานก็ขึ้นอยู่กับกิโลไบต์ต่อวินาที แล้วเปรียบเทียบกับ ค่าการใช้งานสูงสุดของเครือข่ายเป็นเปอร์เซนต์ ไม่ว่าจะเป็นเครือข่ายแบบอีเทอร์เนต (10 เมกกะบิตต่อวินาที) หรือ โทเคนริง (4 หรือ 16 เมกกะบิตต่อวินาที)

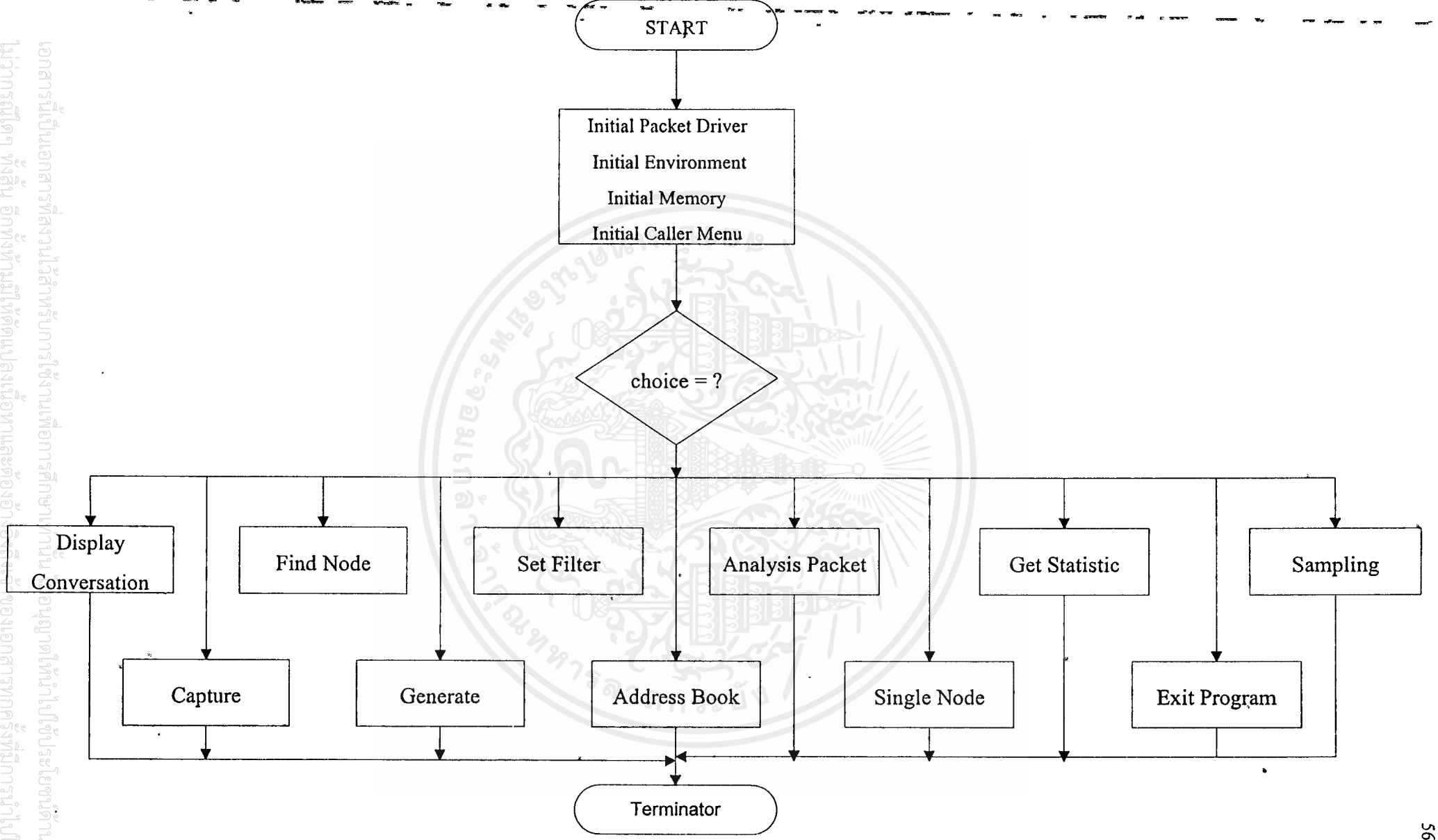
**4.4.4.5 โฮสที่กำลังใช้งานอยู่มากที่สุด**

ควรทราบถึงการใช้งานโฮสบนเครือข่าย การดูแลการใช้งานโฮสจะช่วยในการแบ่งเบาภาระการทำงานระหว่างโฮส ในการวัดการใช้งานของโฮสควรจะต้องดูจำนวนแพ็กเกจต่อวินาทีจากโฮสเป็นระยะเวลาหลายวันหรือหลายสัปดาห์ ถ้าคุณพบว่ามีจำนวนผู้ใช้มากในหนึ่งเครือข่าย โดยการเฝ้าดูเซิร์ฟเวอร์ (Server) ที่มีการใช้งานมากที่สุด คุณอาจจะหลีกเลี่ยงการใช้งานมากเกินไปได้ ถ้าประสิทธิภาพลดลงแต่กลับมีการโปรเซสแพ็กเกจอยู่ อาจเป็นไปได้ที่เกิดคอขวดที่การ์ดเชื่อมต่อของเซิร์ฟเวอร์

## 4.5 การสร้างโปรแกรม

ส่วนเริ่มแรกของโปรแกรมคือส่วนเมนูจะเป็นตัวเลือกเพื่อไปยังส่วนต่าง ๆ ของโปรแกรม ซึ่งมีส่วนต่าง ๆ ที่ใช้ในโปรแกรมดังนี้

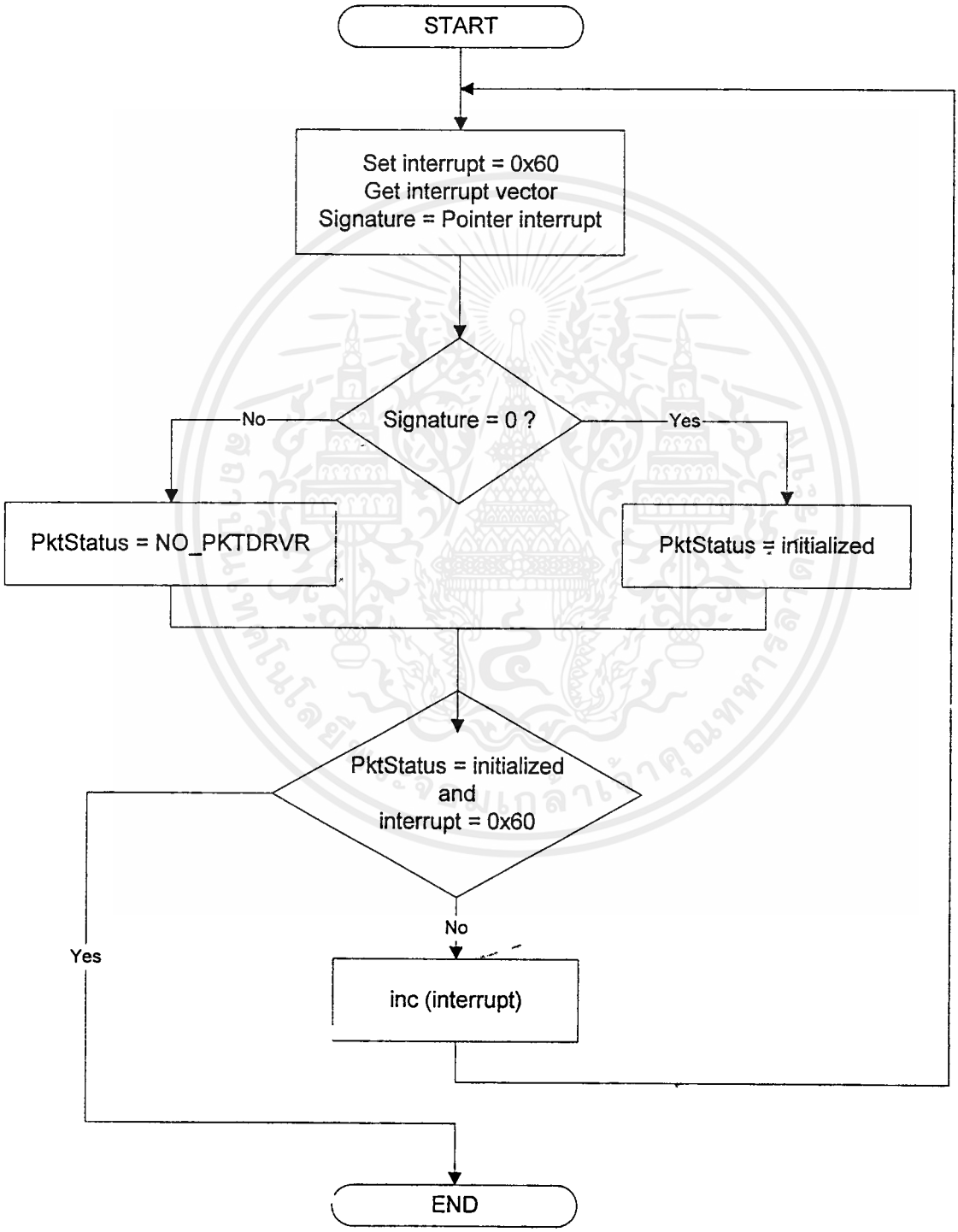
- แสดงการสนทนาระหว่างโฮสุ (Display Conversation)
- รับและแสดงส่วนของเน็ตเวิร์ค (Get Statistic)
- ตรวจสอบและค้นหาเครื่อง (Find Node)
- เก็บแพ็กเกจ (Capture)
- ตั้งค่าการกรองแพ็กเกจ (Set Filter)
- สร้างแพ็กเกจเพื่อทดสอบเตเวิร์ค (Generate Packet)
- วิเคราะห์แพ็กเกจ (Analysis Packet)



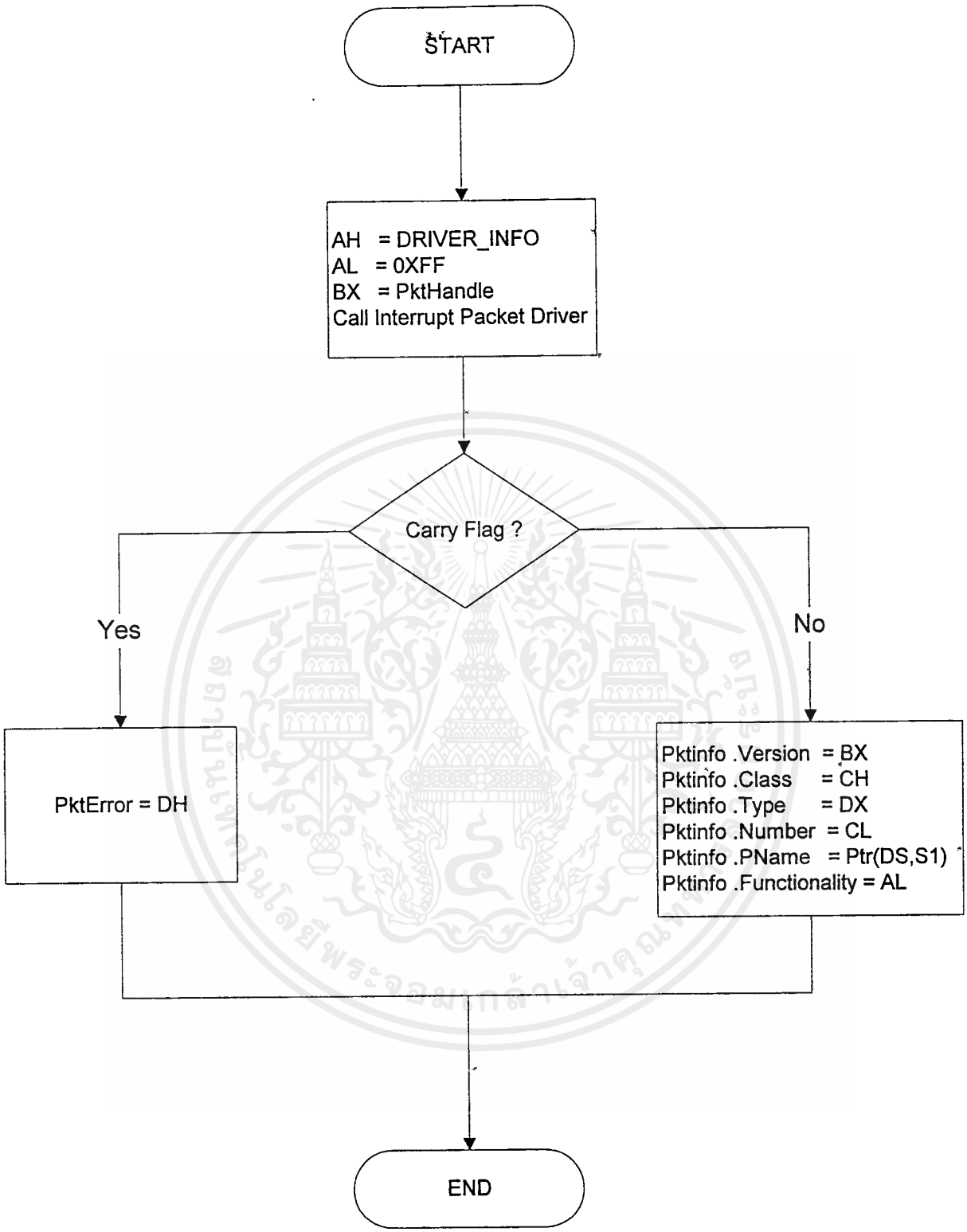

รูปที่ 4.2 ขั้นตอนการทำงานของเมนู

### 4.5.1 โมดูลเริ่มต้นเมนู

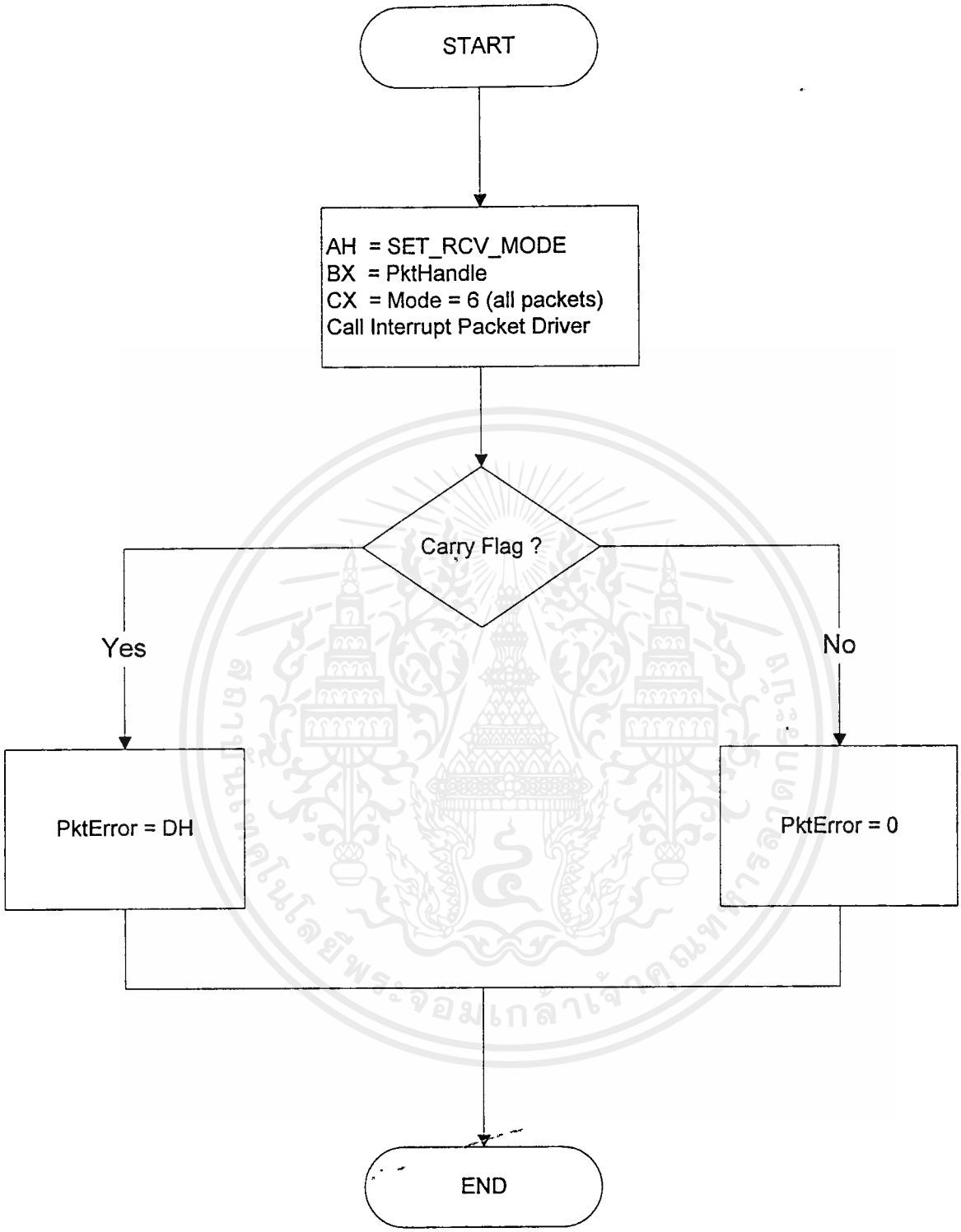
โดยเริ่มแรกของโปรแกรม ก็กำหนดการทำงานในส่วนต่าง ๆ เช่น ส่วนของแพ็กเกจไดรเวอร์ ตัวแปร หน่วยความจำ และเมนู แล้วจึงทำโปรเซสต่าง ๆ ของส่วนโปรแกรม ซึ่งในส่วนการกำหนดค่าของแพ็กเกจไดรเวอร์ โดยการค้นหาบริการอินเทอร์รัปที่มีให้ (User Interface Interrupt Service) แล้วจึงตั้งค่าการทำงานของแพ็กเกจไดรเวอร์ต่าง ๆ เช่น วิธีการรับแพ็กเกจ การทำค่าแฮนเดิลของบริการ และฟังก์ชั่นที่สนับสนุน



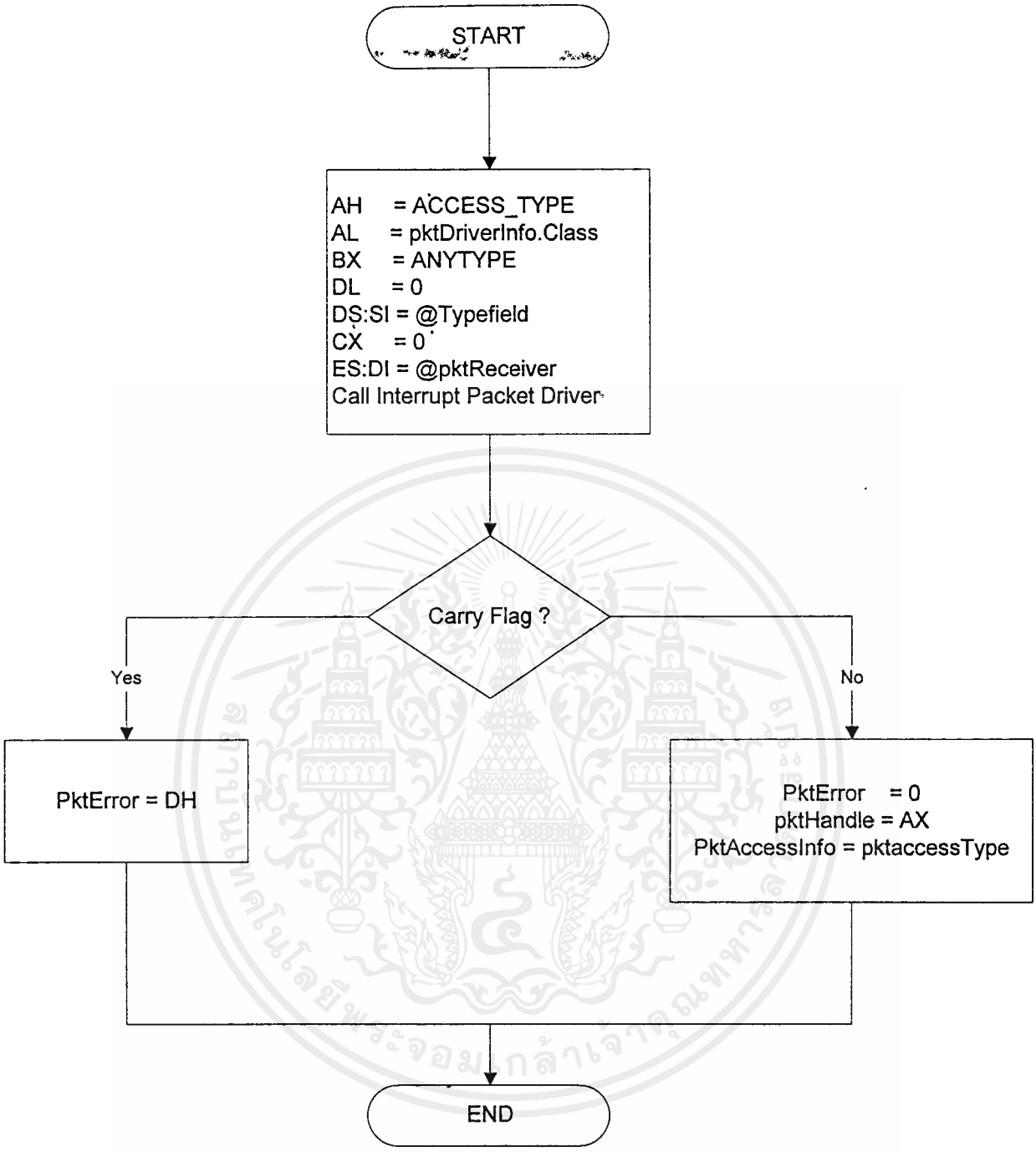

รูปที่ 4.3 ขั้นตอนการหาแพ็กเกจไดรเวอร์




รูปที่ 4.4 ขั้นตอนการรับข้อมูลของแพ็กเกจไดรเวอร์




รูปที่ 4.5 ขั้นตอนการตั้งค่าวิธีการรับแพ็คเกจ

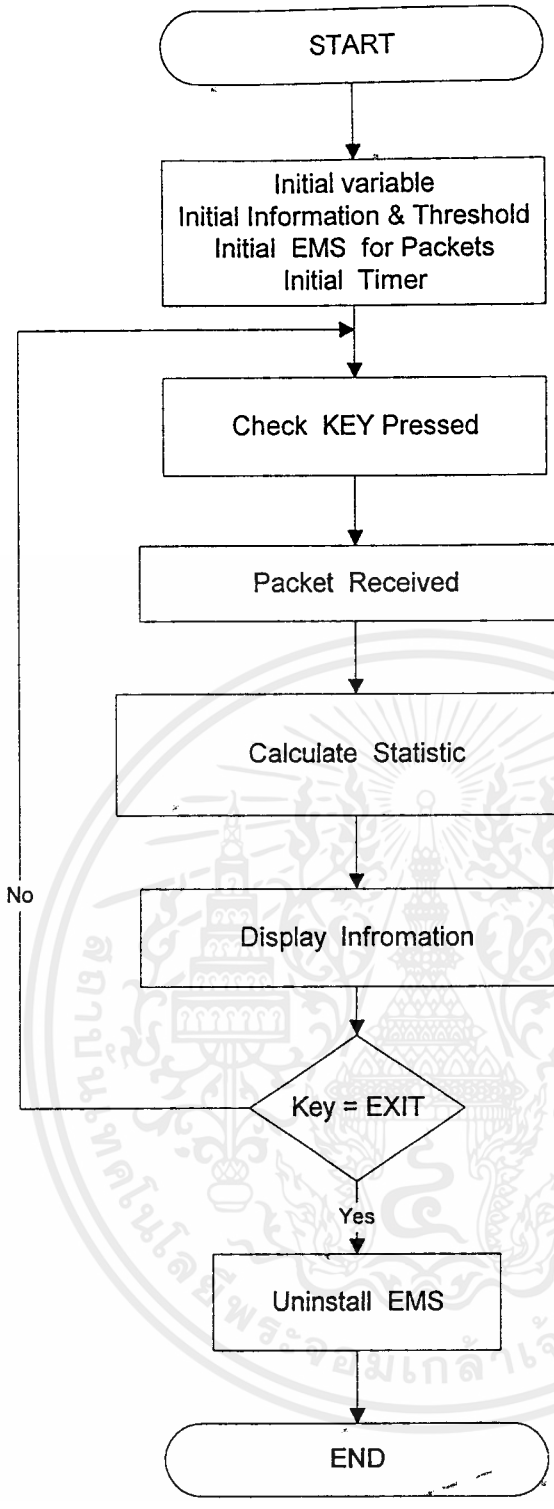



รูปที่ 4.6 ขั้นตอนการตั้งค่าของแพ็คเกจไดรเวอร์

### 4.5.2 โมดูลแสดงการสนทนา

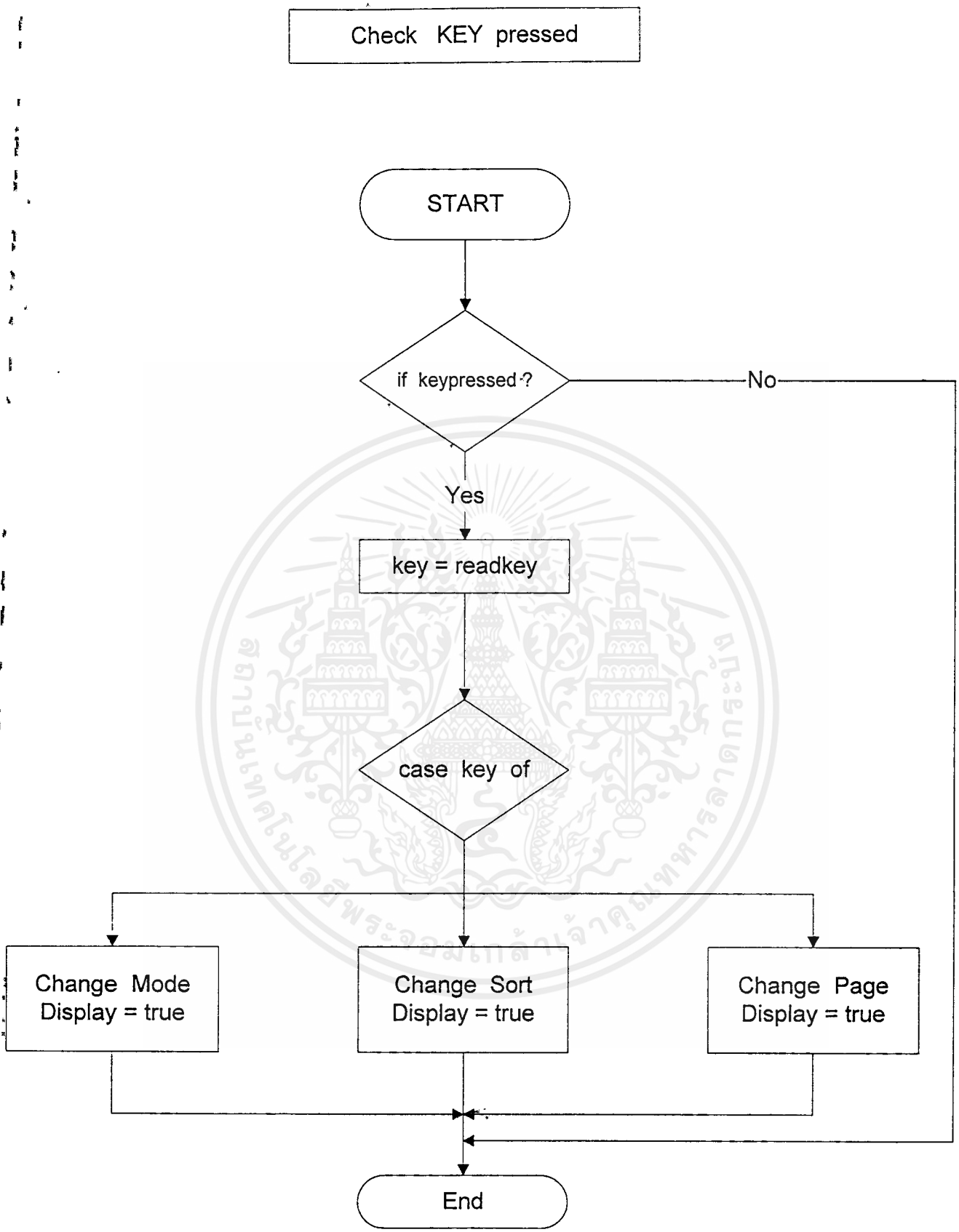
แสดงการสนทนาระหว่างโฮสด้วยกันโดยมีรายละเอียดส่วนหลัก ๆ ของโมดูลนี้ดังนี้

- ตรวจสอบคีย์
- รับแพ็กเกจ
- คำนวณสถิติ
- แสดงข้อมูล

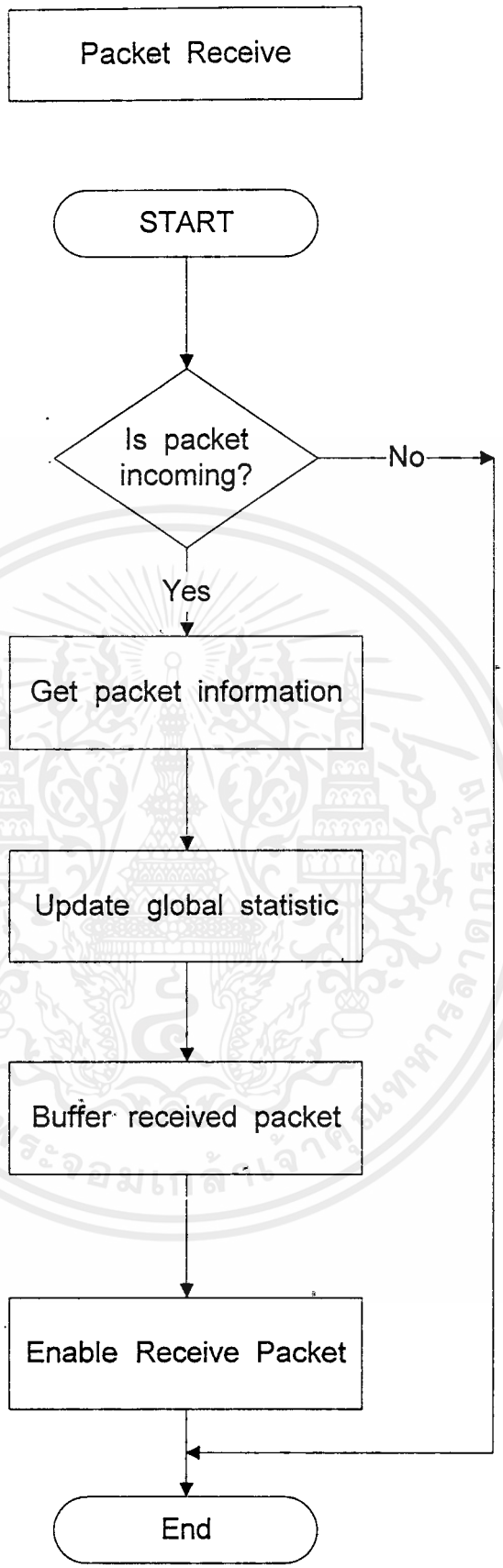



รูปที่ 4.7 ขั้นตอนการทำงานของโมดูลแสดงการสนทนา

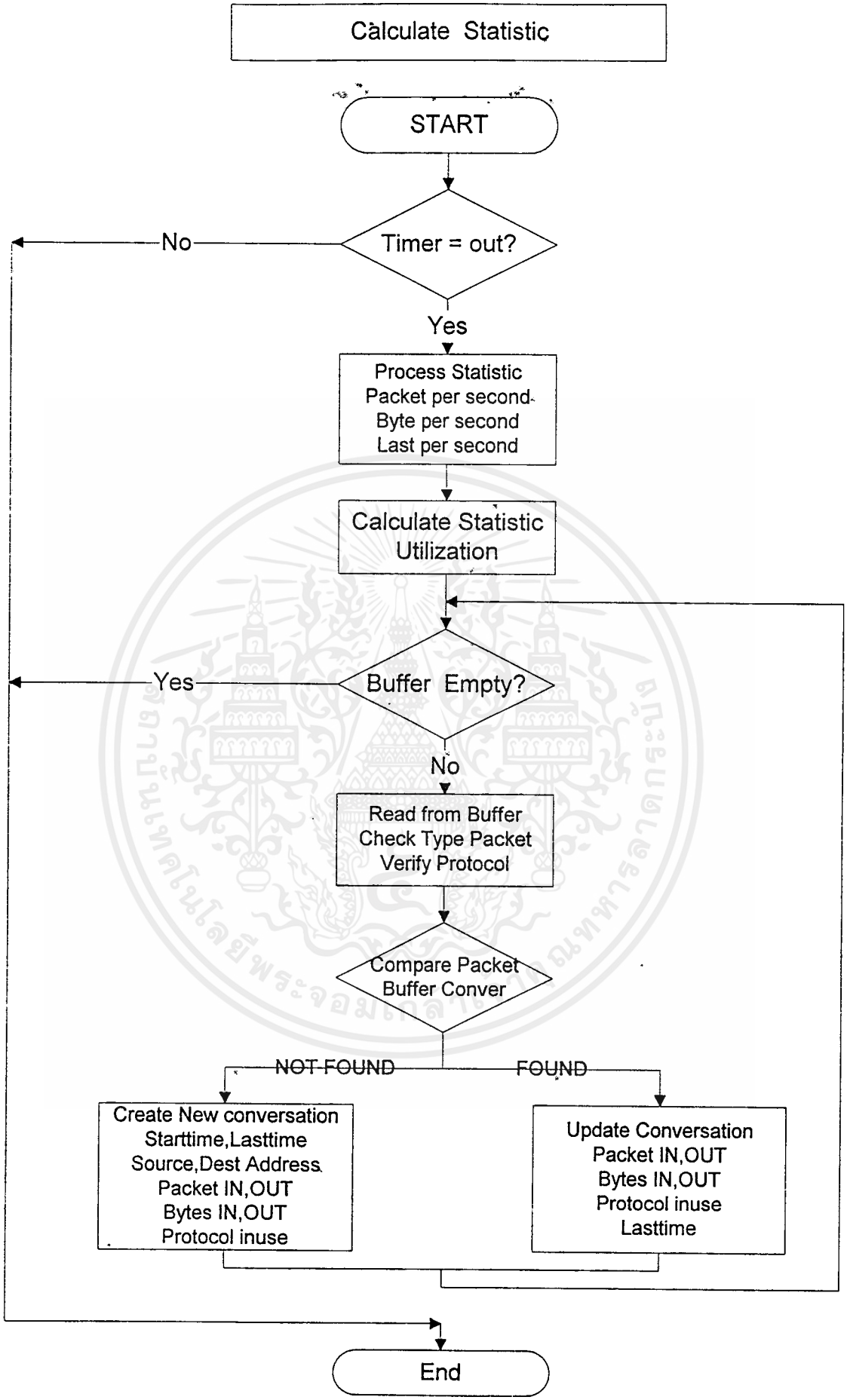
เริ่มต้นก็รับข้อมูลจากแพ็กเกจมาก่อนในช่วงระยะเวลาหนึ่ง โดยเก็บข้อมูลไว้ในบัฟเฟอร์เพื่อให้มีเวลาในการทำงานส่วนของการคำนวณให้มีประสิทธิภาพ และหลังจากนั้นก็แสดงผล



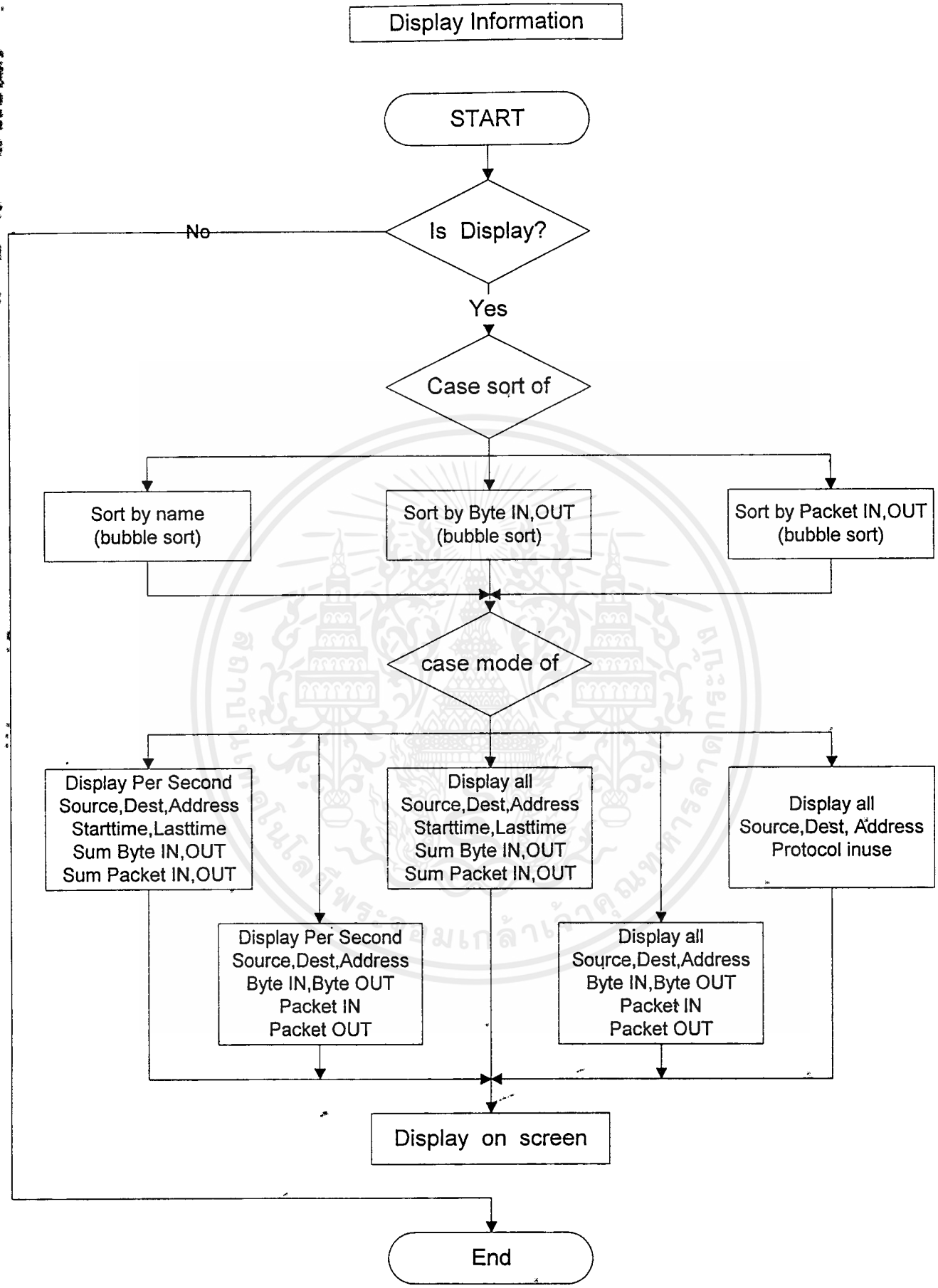

รูปที่ 4.8 ขั้นตอนการทำงานของโมดูลแสดงการสนทนาส่วนรับคีย์




**รูปที่ 4.9 ขั้นตอนการทำงานของโมดูลแสดงการสนทนาส่วนรับแพ็กเกจ**



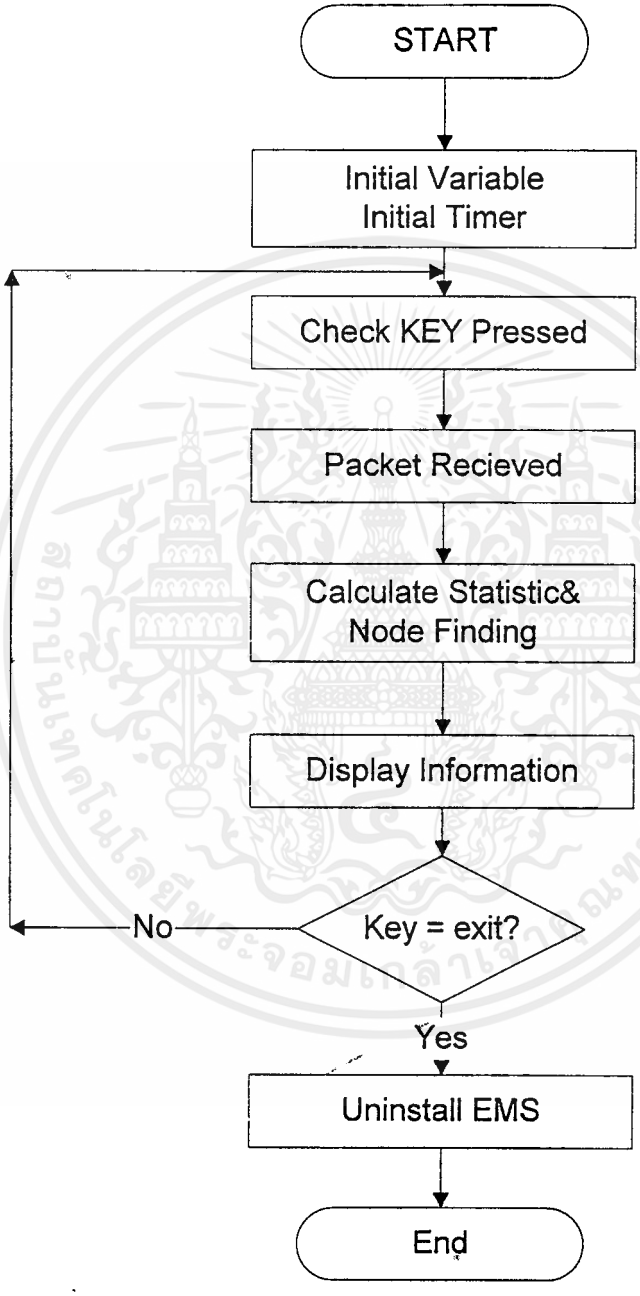

รูปที่ 4.10 ขั้นตอนการทำงานของโมดูลแสดงการสนทนาส่วนคำนวณสถิติ



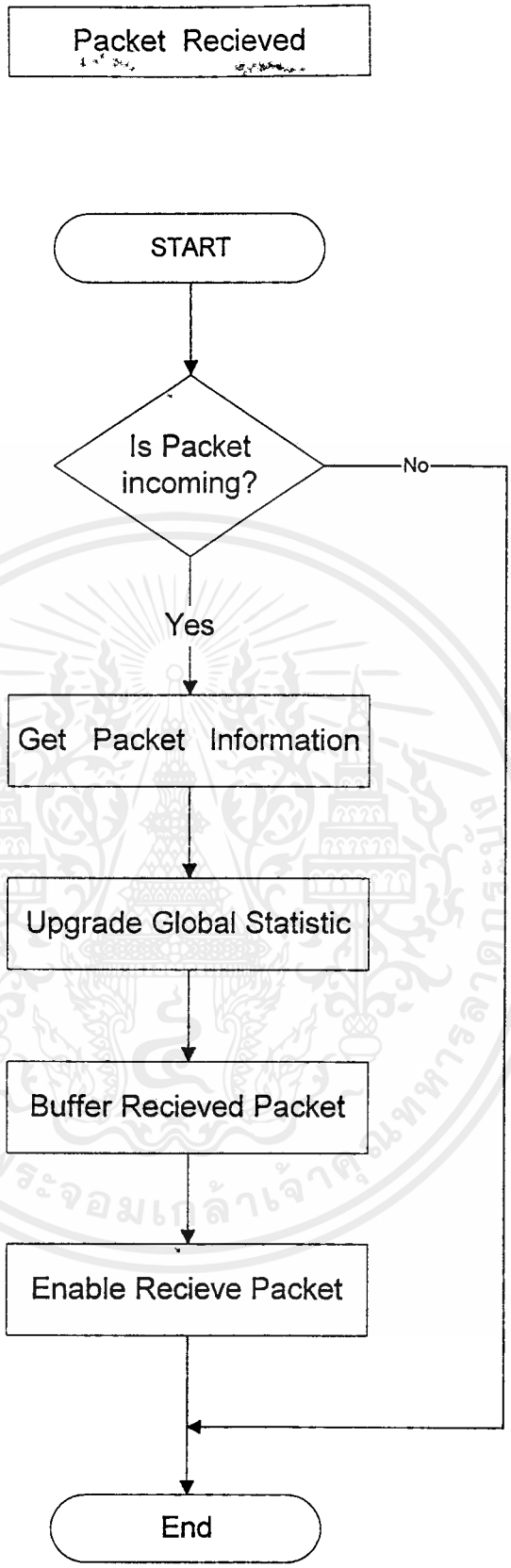

รูปที่ 4.11 ขั้นตอนการทำงานของโมดูลแสดงการสนทนาส่วนแสดงผล

### 4.5.3 โมดูลค้นหาโหนด

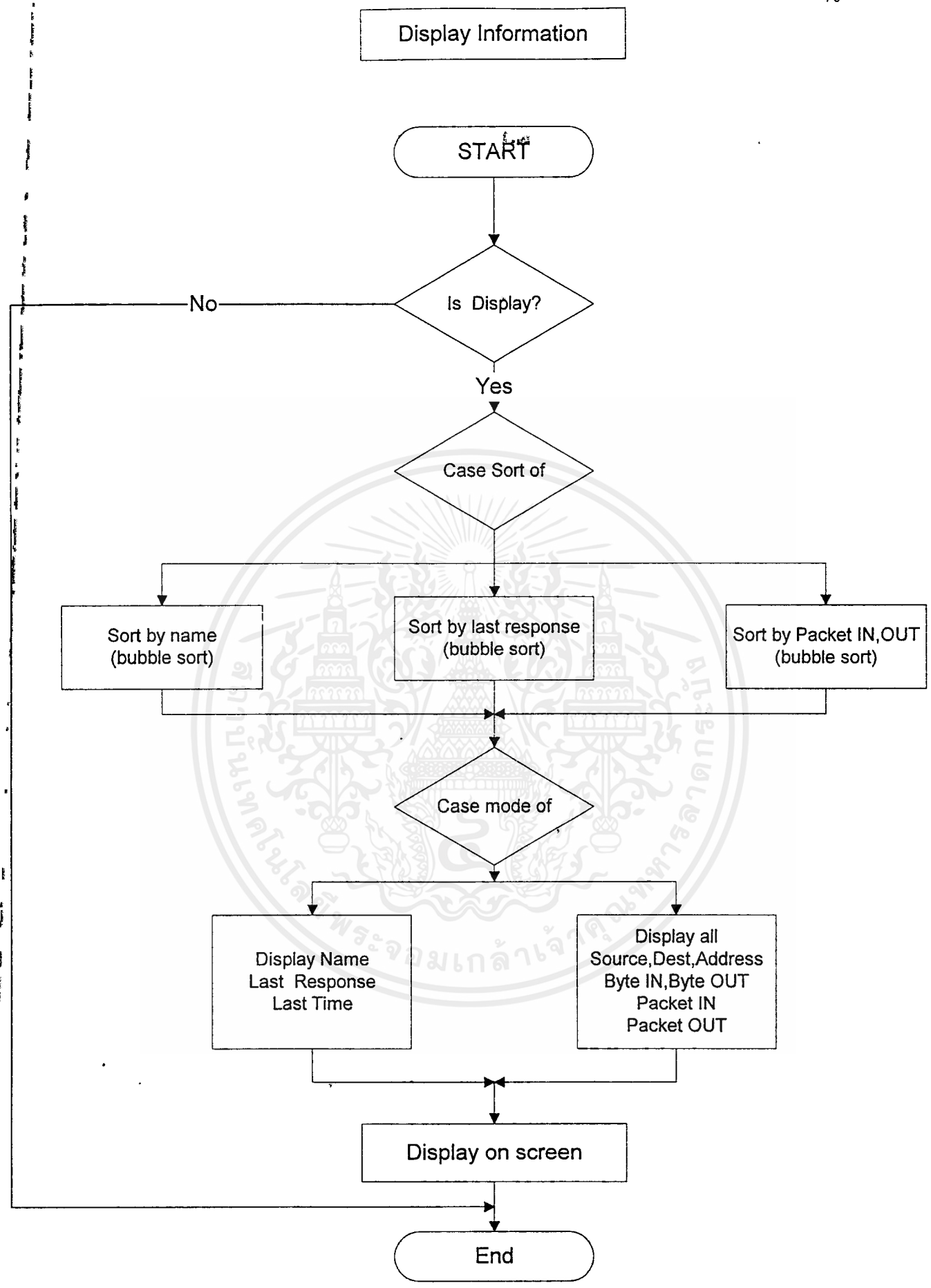
ในส่วนนี้จะแบ่งช่วงการค้นหาโหนดเป็น 2 ช่วงที่ค้นหาจากแพ็กเกจที่วิ่งอยู่ และจากข้อมูลที่เก็บไว้ ในแบบที่หนึ่งนั้นจะใช้วิธีการรับข้อมูลจากแพ็กเกจแล้วค่อยแสดงข้อมูลต่าง ๆ ในเน็ตเวิร์คจากแพ็กเกจที่ได้ในรูปแบบที่สองนั้นจะมีรายชื่อของโฮสไว้แล้วใช้โปรโตคอล ARP และ ICMP ช่วยในการค้นหาโฮส โดยไปถามว่ายังอยู่ดีหรือไม่



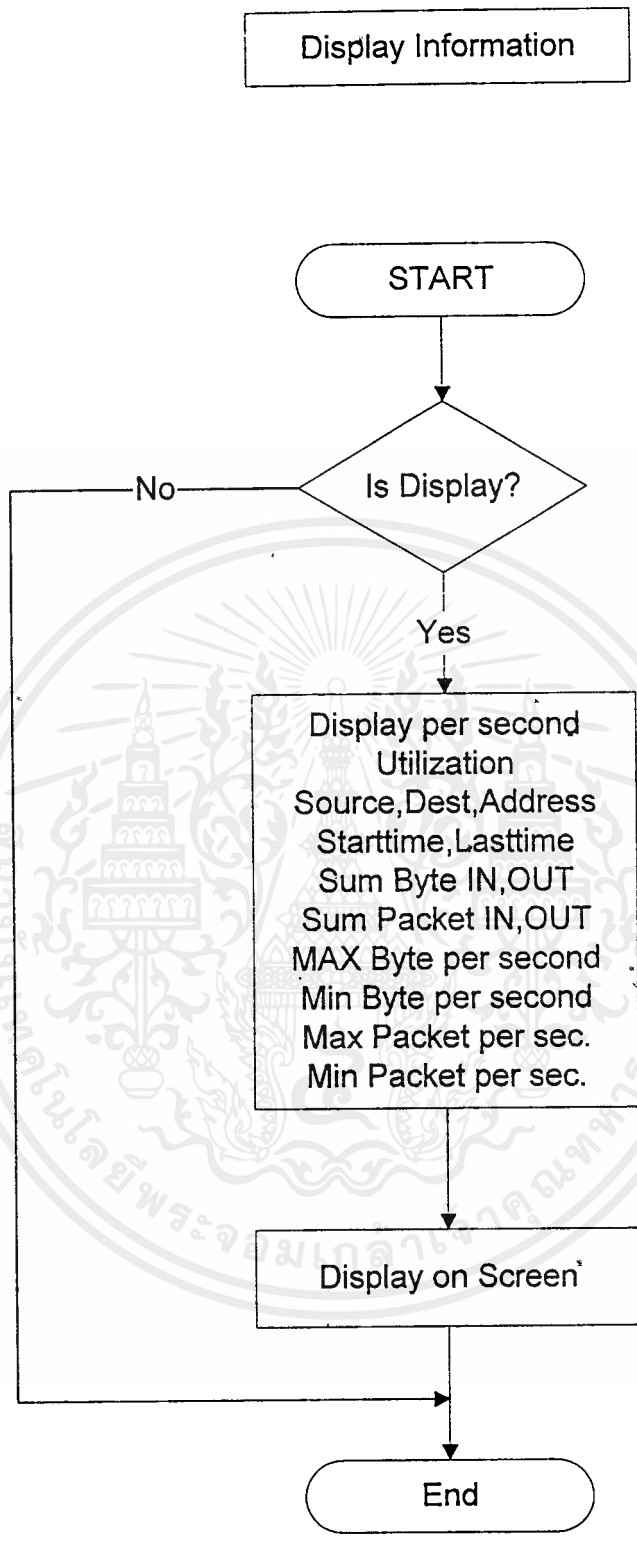

รูปที่ 4.12 ขั้นตอนการทำงานของโมดูลค้นหาโหนด




รูปที่ 4.14 ขั้นตอนการทำงานของโมดูลค้นหาโหนดส่วนรับแพ็กเกจ



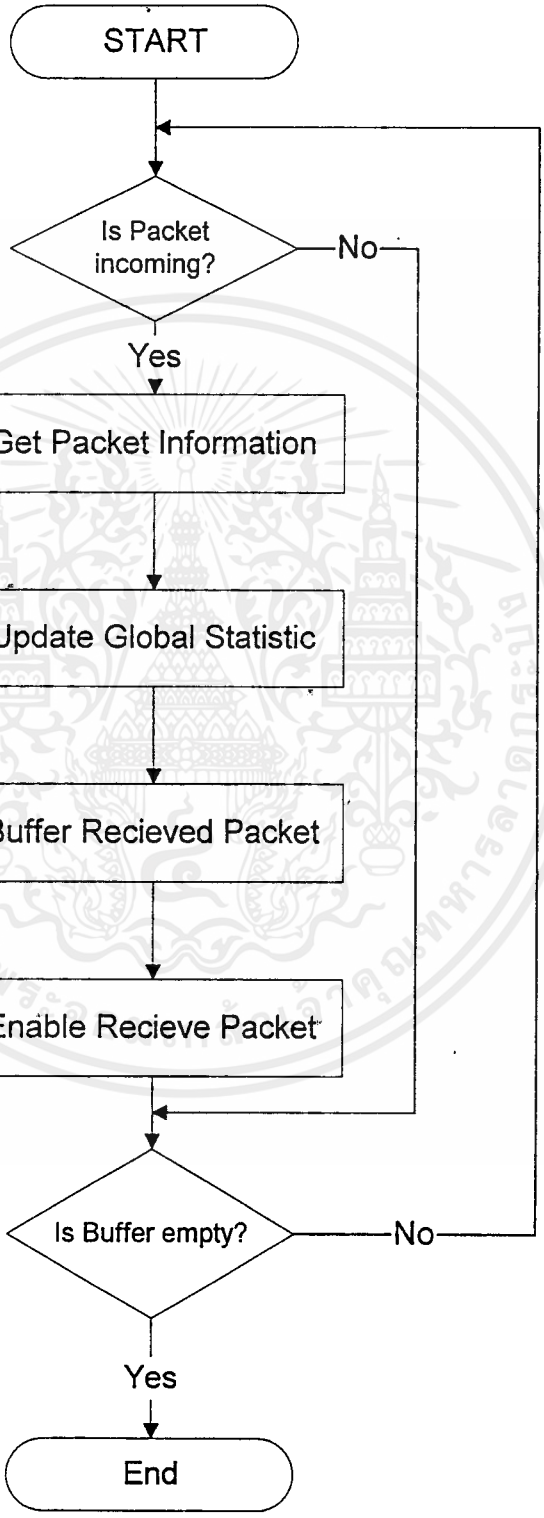

รูปที่ 4.16 ขั้นตอนการทำงานของโมดูลค้นหาโหนดส่วนแสดงผล




รูปที่ 4.19 ขั้นตอนการทำงานของโมดูลคำนวณสถิติส่วนแสดงผล

### 4.5.5 โมดูลเก็บแพ็กเกจ

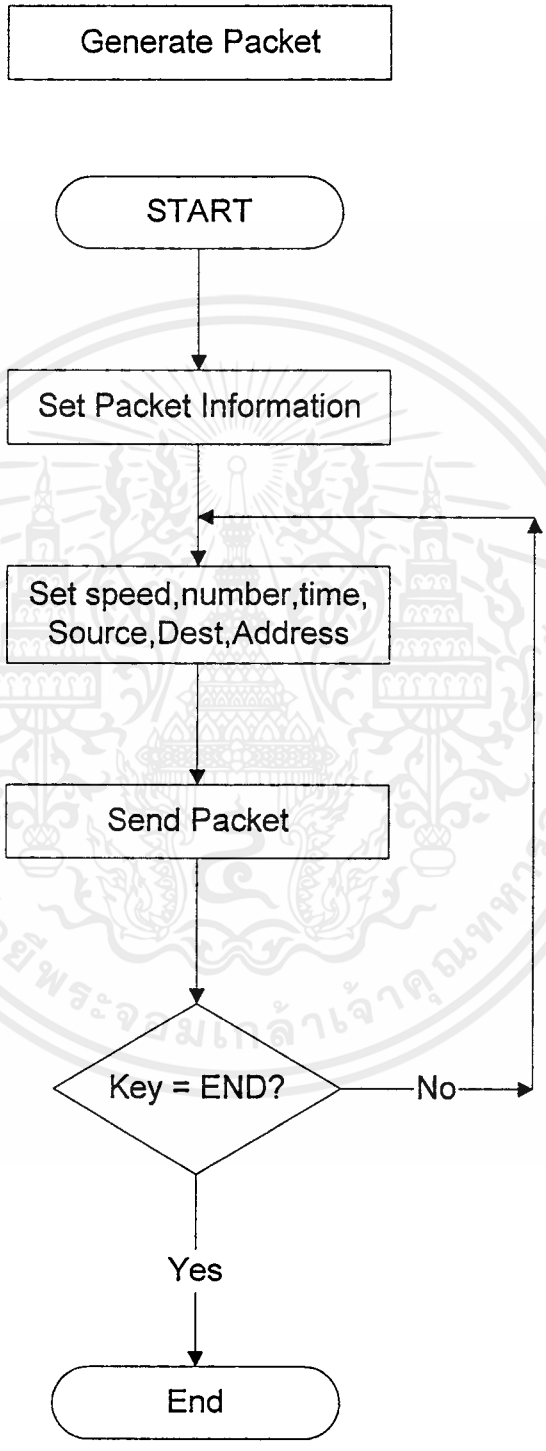
ส่วนที่เก็บข้อมูลของแพ็กเกจที่วิ่งในเน็ตเวิร์คโดยเก็บข้อมูลด้วยกันอยู่ 2 รูปแบบ คือเก็บลงในหน่วยความจำหรือ ฮาร์ดดิสก์




รูปที่ 4.20 ขั้นตอนการทำงานของโมดูลเก็บแพ็กเกจ

### 4.5.6 โมดูลสร้างแพ็กเกจ

ในส่วนนี้จะสร้างแพ็กเกจตามรูปแบบและขนาดที่กำหนด โดยส่งข้อมูลไปในเครือข่ายเพื่อทดสอบระบบ โดยสามารถเลือกการส่งข้อมูลได้โดยกำหนดจำนวนแพ็กเกจ เวลา ความเร็วในการส่ง รวมถึง ต้นทางและปลายทางที่ต้องการจะส่งด้วย

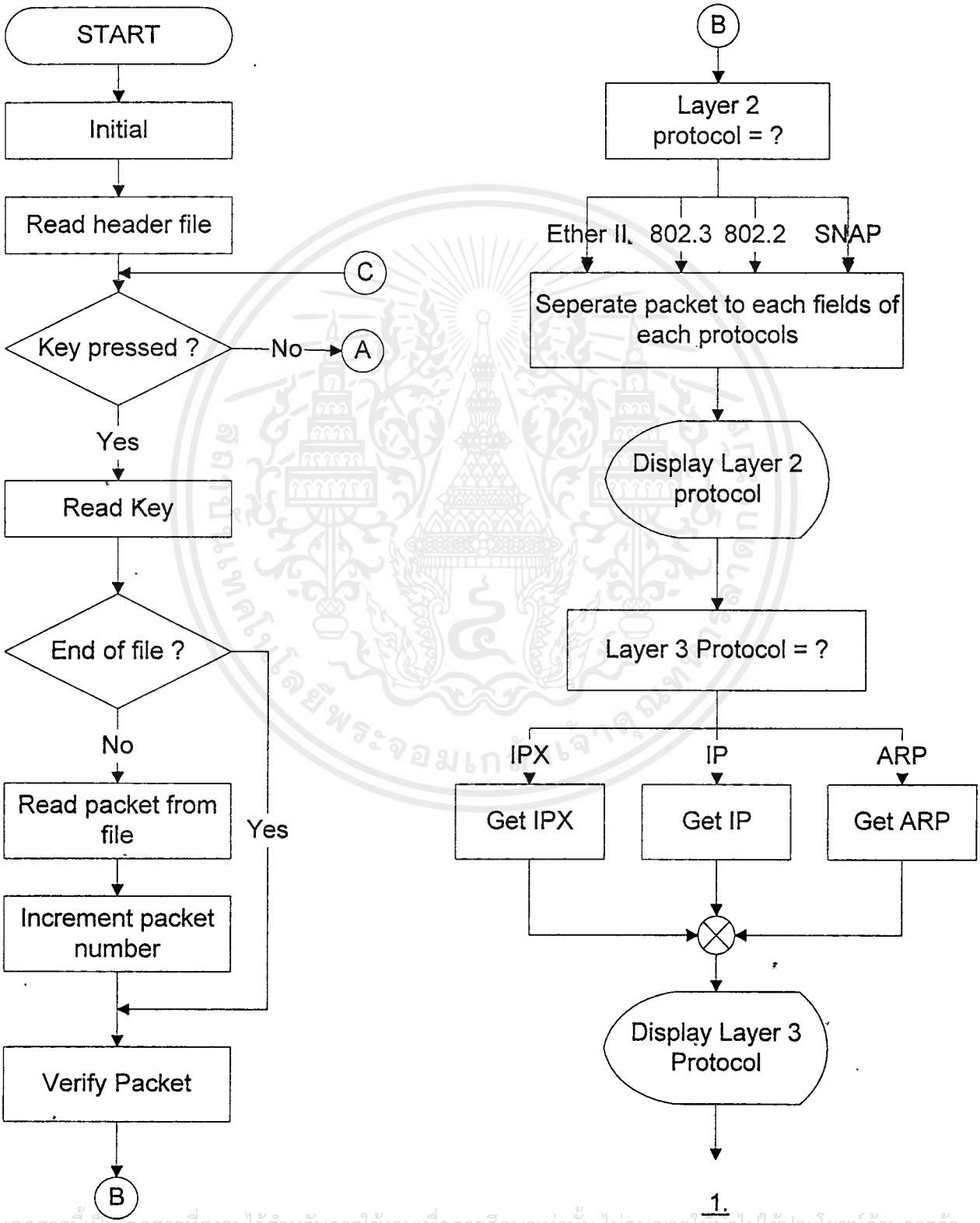



รูปที่ 4.21 ขั้นตอนการทำงานของโมดูลสร้างแพ็กเกจ

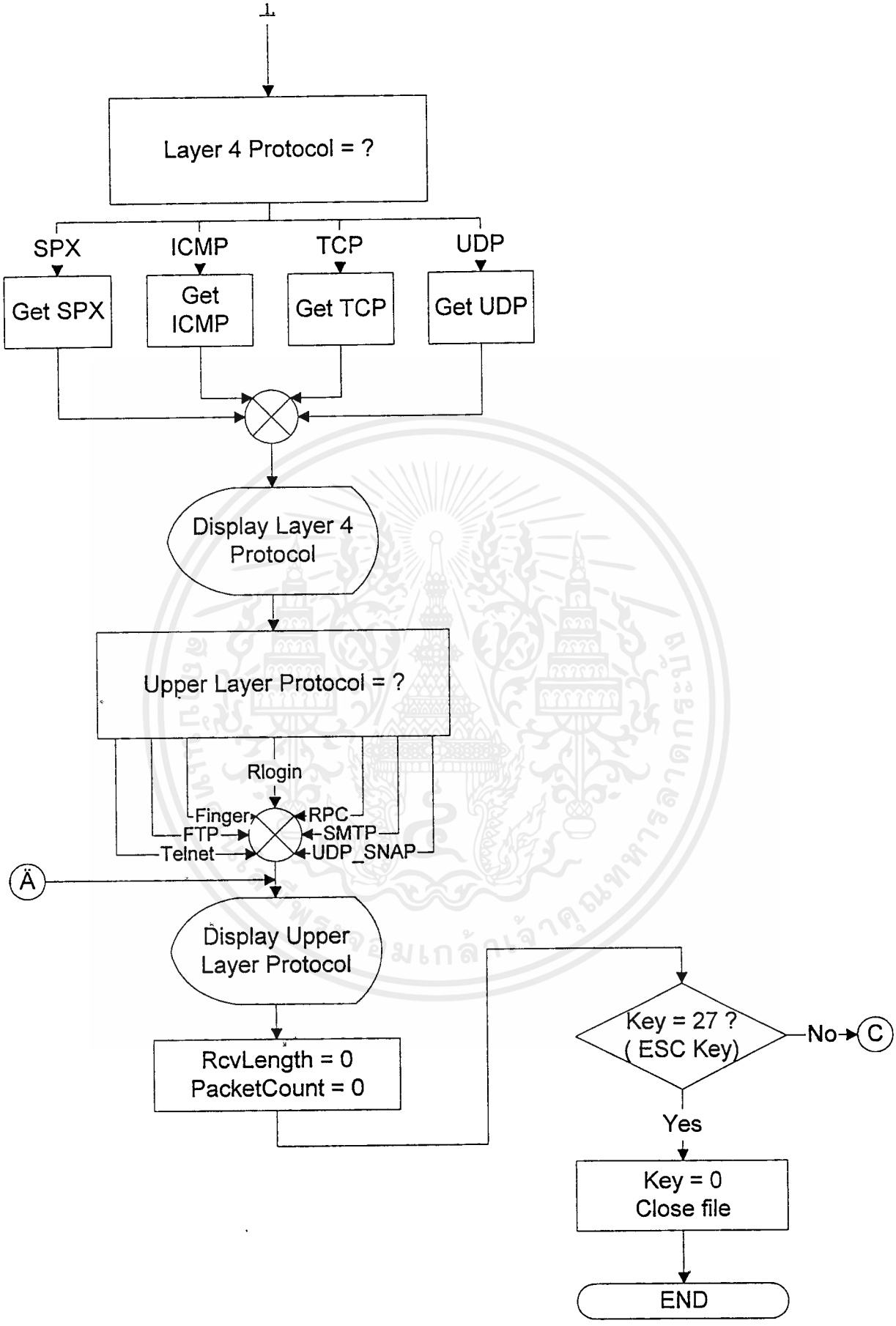
### 4.5.7 โมดูลวิเคราะห์แพ็กเกจ

ในส่วนของโมดูลนี้มีหน้าที่การทำงานในส่วนต่าง ๆ เกี่ยวกับแพ็กเกจเพื่อใช้ในการวิเคราะห์ถึงข้อมูลต่าง ๆ ได้แบ่งเป็นโมดูลย่อย ๆ ได้ดังนี้

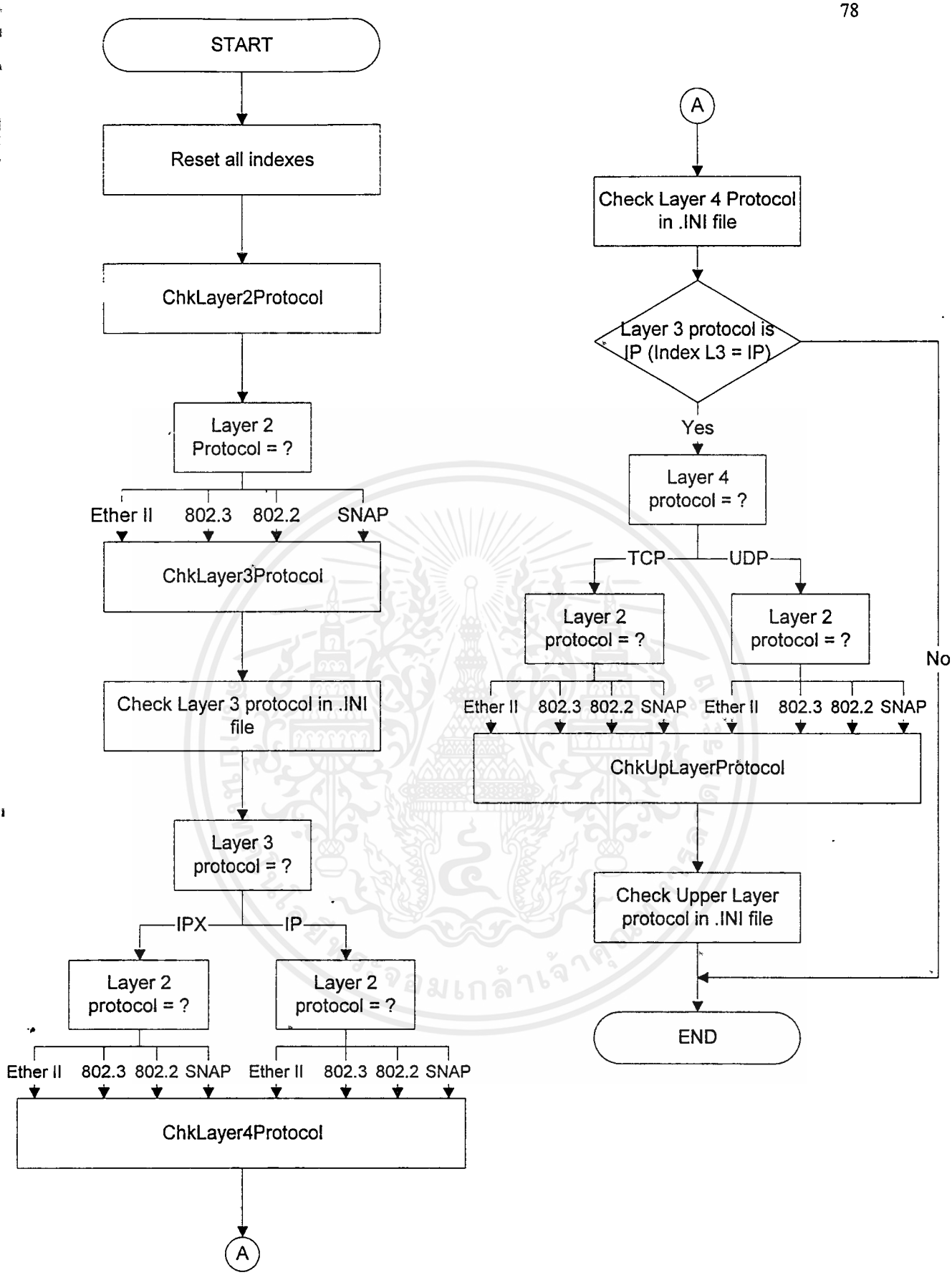
- วิเคราะห์โปรโตคอล
- วิเคราะห์ขนาดเฟรม



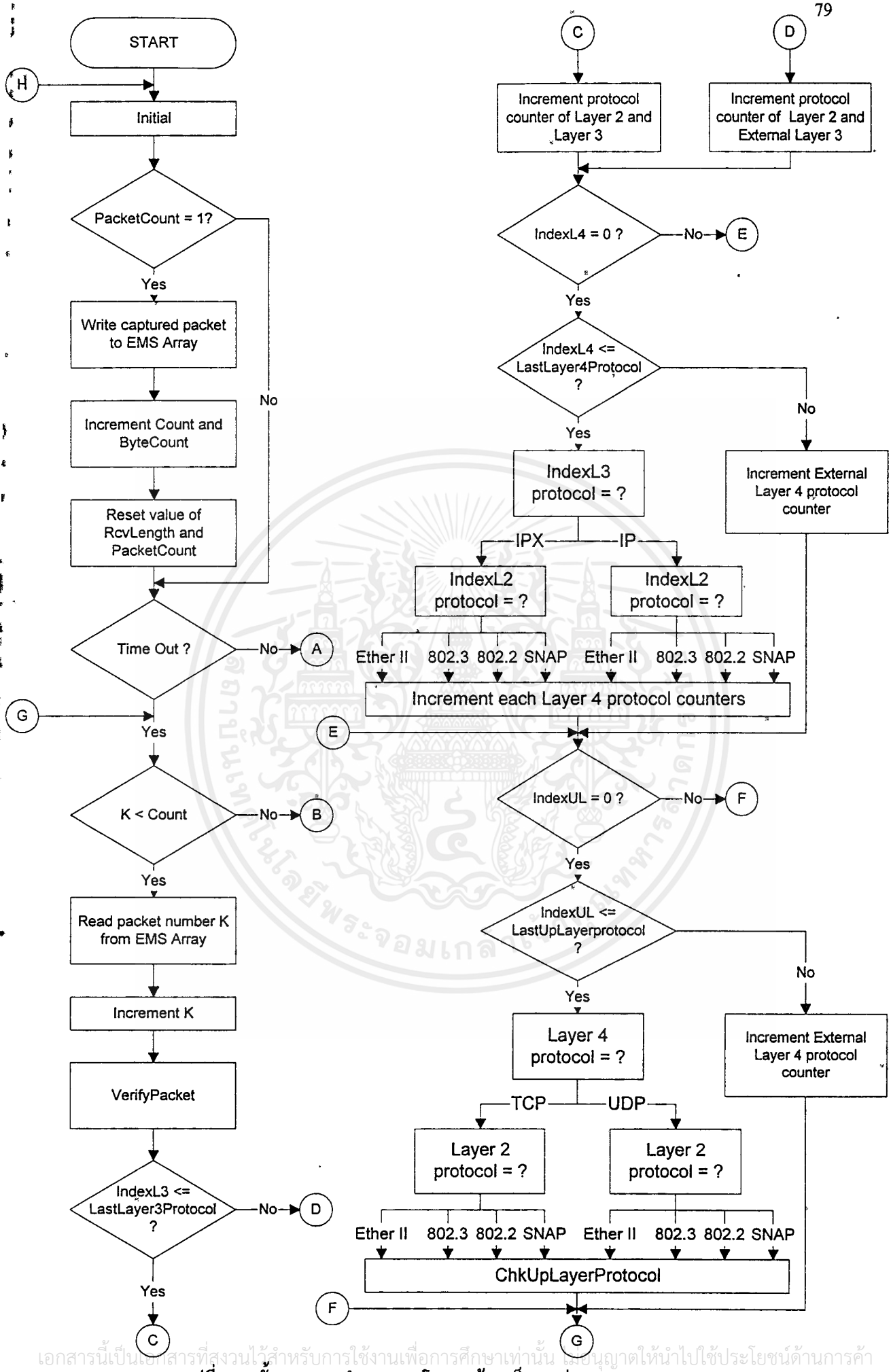

รูปที่ 4.22 ขั้นตอนการทำงานของโมดูลวิเคราะห์แพ็กเกจส่วนที่ 1



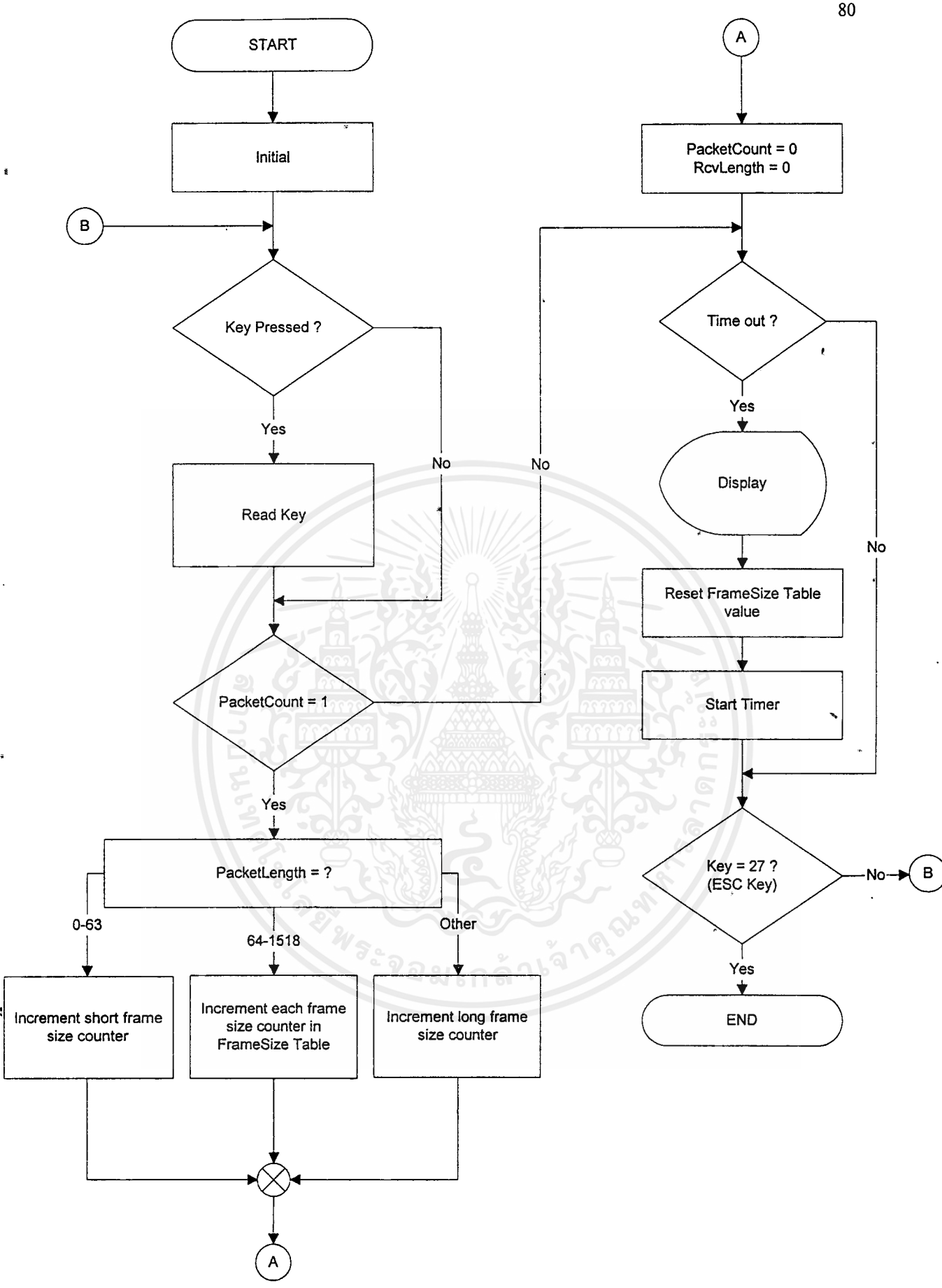

รูปที่ 4.23 ขั้นตอนการทำงานของโมดูลวิเคราะห์แพ็กเกจส่วนที่ 2




รูปที่ 4.24 ขั้นตอนการทำงานของโมดูลตรวจสอบแพ็คเกจ



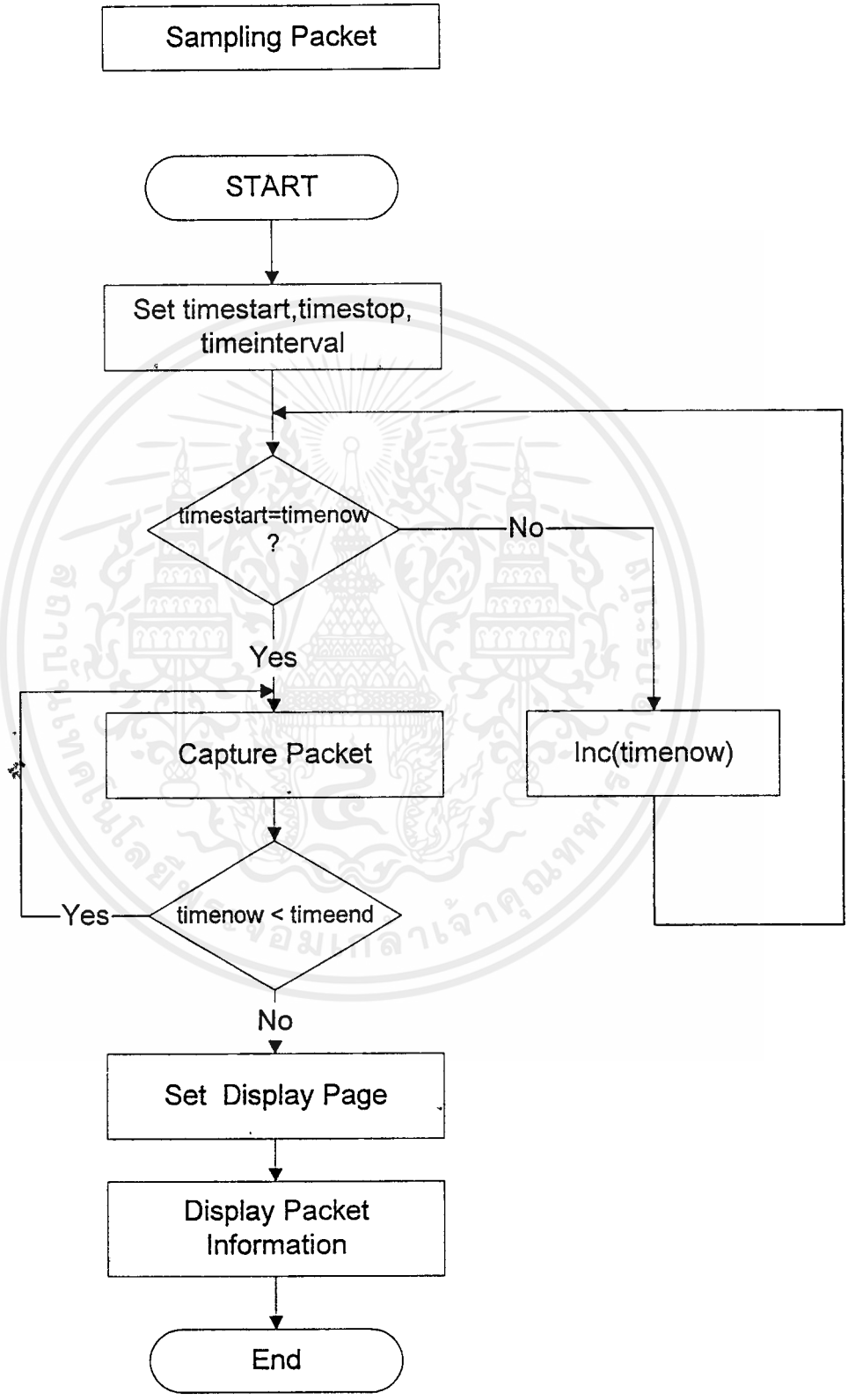

รูปที่ 4.25 ขั้นตอนการทำงานของโมดูลสร้างแพ็กเกจส่วนแสดงผล




**รูปที่ 4.26 ขั้นตอนการทำงานของโมดูลวิเคราะห์แพ็กเกจส่วนการกระจายเฟรม**

### 4.5.8 โมดูลสุ่มเก็บข้อมูลเครือข่าย

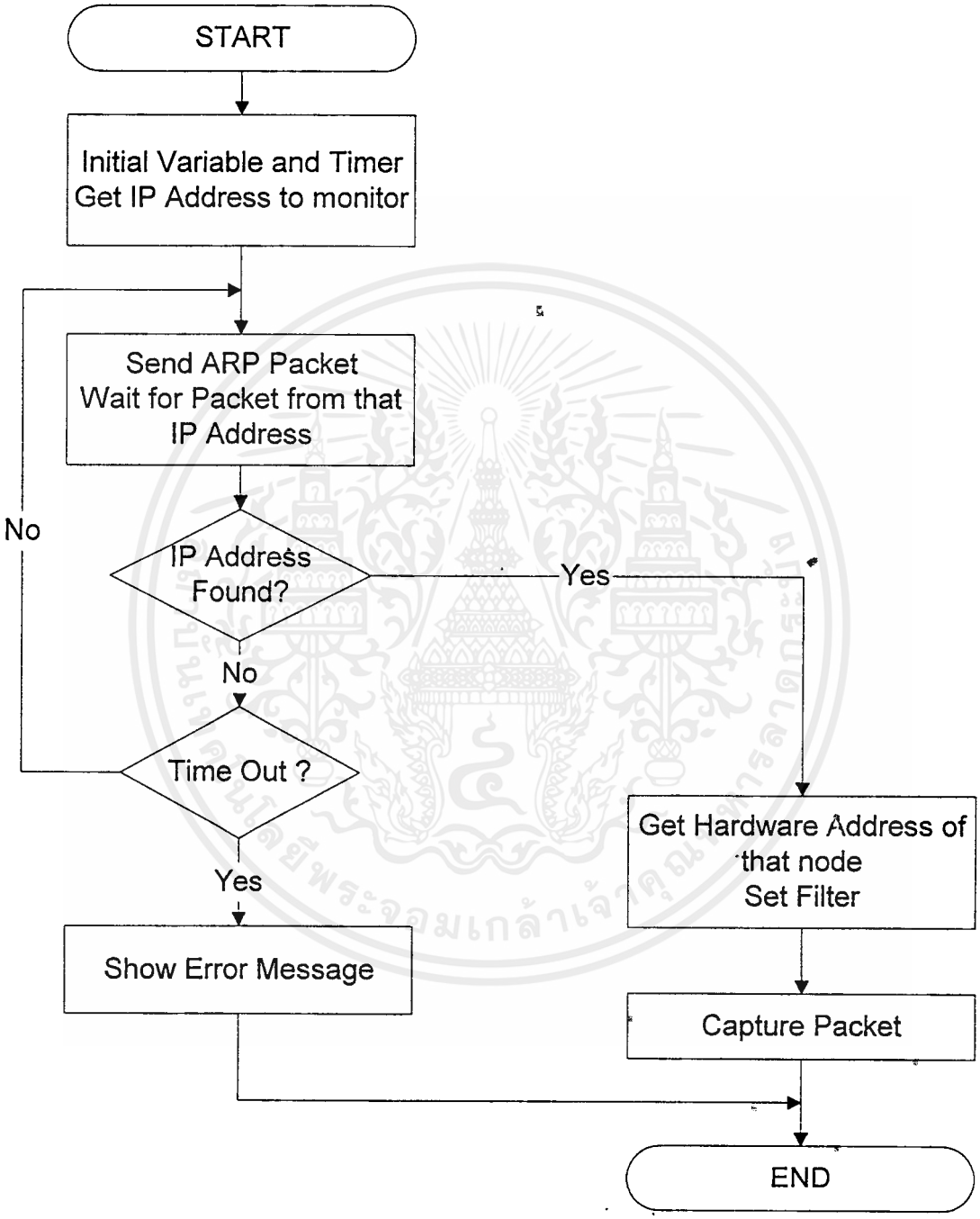
ใช้ในการสุ่มเก็บข้อมูลในช่วงระยะเวลาหนึ่ง แล้วนำมาแสดงผล โดยสามารถระบุช่วงเวลาที่เก็บ และ ประเภทของข้อมูลที่ต้องการเก็บได้




รูปที่ 4.27 ขั้นตอนการทำงานของโมดูลส่วน Sampling Packet

### 4.5.9 โมดูลเฝ้าดูเฉพาะเครื่อง

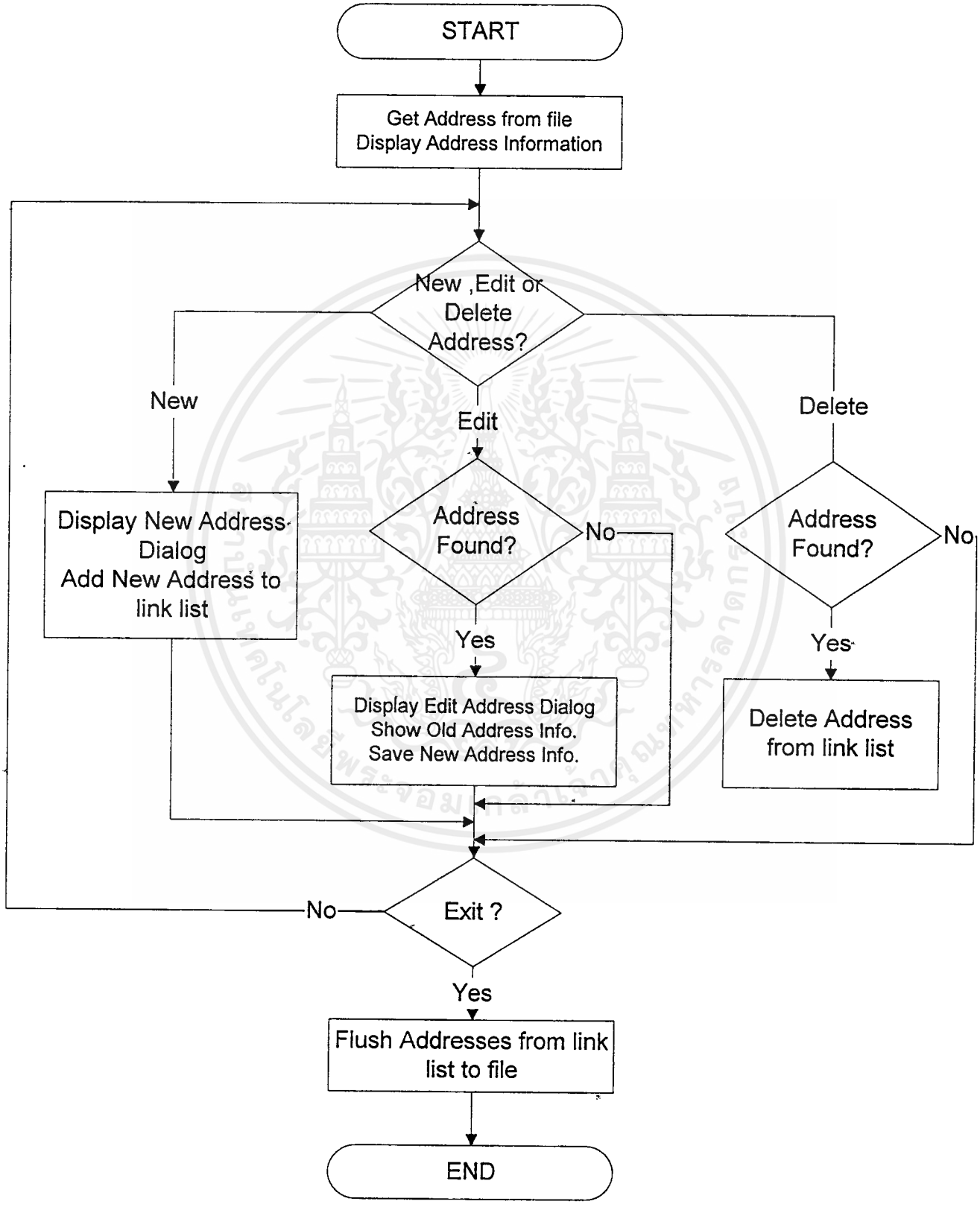
ใช้ในการตรวจสอบเฉพาะโหนดใดโหนดหนึ่ง ในเครือข่าย เพื่อแสดงถึงข้อมูลภายในแพ็กเกจที่โหนดนั้นรับและส่ง โดยการระบุ IP Address ของโหนดนั้น




รูปที่ 4.28 ขั้นตอนการทำงานของโมดูล ตรวจสอบโหนดเดี่ยว (Single Node)

### 4.5.10 โมดูลสมุดบันทึก

เพื่อให้ผู้ใช้สามารถจดจำโหนดใด ๆ ใน เครือข่ายได้ โดยการบันทึกชื่อที่จำง่าย และเลขฮาร์ดแวร์แอดเดรส ไอพีแอดเดรส รวมทั้งคำอธิบายได้อีกด้วย โดยสามารถเพิ่มเติม แก้ไข และลบข้อมูลเก่าได้




รูปที่ 4.29 ขั้นตอนการทำงานของโมดูล Address Book

# บทที่ 5

# การทดลองและผลการทดลอง

## 5.1 การดำเนินงานในภาคเรียนที่ 1/2540

รายละเอียดและขั้นตอนการดำเนินงานมีดังนี้

1. ศึกษาโครงสร้าง เฟรมชนิดอีเทอร์เนต และโปรโตคอลต่าง ๆ
2. ศึกษาการทำงาน การ์ดเชื่อมต่อเน็ตเวิร์ค และ ndis ไดรเวอร์
3. ศึกษาโครงสร้าง ทีซีพี/ไอพี
4. ศึกษาการทำงาน และวิธีการของโปรแกรมตรวจสอบเน็ตเวิร์คอื่น ๆ
5. ศึกษาวิธีการดักจับข้อมูล เก็บข้อมูล วิเคราะห์ข้อมูล
6. ออกแบบระบบ
7. จัดทำเอกสาร

## 5.2 ผลการดำเนินงาน

ศึกษาขั้นตอนการทำงานของโปรแกรมต่าง ๆ ที่เกี่ยวข้องกับการดักจับข้อมูล ซึ่งได้เลือกวิธีการแบบ เฝ้าดูภายนอก นั้นคือการดักจับข้อมูลที่วิ่งผ่าน การ์ดเชื่อมต่อเครือข่ายโดยตรง ซึ่งยังต้องตรวจสอบถึงความสามารถของฮาร์ดแวร์ และซอฟต์แวร์ ว่าสามารถดักจับโค้ดได้ทันหรือไม่ ขั้นต่อมาได้ศึกษารูปแบบเฟรมต่าง ๆ ที่สำคัญได้แก่ ไอพี เออาร์พี ไอซีเอ็มพี ยูดีพี เป็นต้น และศึกษาถึงการทำงานและการเรียกใช้ของ Network Drover Interface Specificationว่าสามารถที่จะใช้ทำงานร่วมกับ Project นี้ได้หรือไม่

## 5.3 การดำเนินงานในภาคเรียนที่ 2/2540

รายละเอียดและขั้นตอนการดำเนินงานมีดังนี้

- ออกแบบซอฟต์แวร์
- พัฒนาส่วน Sampling
- พัฒนาส่วน Address Book
- พัฒนาส่วน Single Node
- รวมระบบพร้อมกับทดสอบการใช้งาน
- จัดทำคู่มือการใช้งาน
- ใช้งานและดูแล

5.3.1 การออกแบบซอฟต์แวร์

แบ่งเป็น

- ออกแบบโครงสร้าง จะทำการออกแบบโครงสร้างของซอฟต์แวร์ว่าประกอบด้วยส่วนใดบ้าง แต่ละส่วนแบ่งเป็นโมดูลย่อย ๆ ใดบ้าง และสร้างโครงสร้างโมดูลต่าง ๆ ขึ้นเพื่อช่วยในการทำความเข้าใจซอฟต์แวร์
- ออกแบบรายละเอียด จะแบ่งซอฟต์แวร์ออกเป็นโมดูลย่อย ๆ และทำการออกแบบแต่ละโมดูลตามหลัก อ๊อบเจคโดยออกแบบบริการต่าง ๆ รวมถึงการอินเทอร์เฟสระหว่างโมดูลด้วย และออกแบบอัลกอริทึม ที่ดีที่สุดแล้ว ออกแบบลงในรายละเอียดของแต่ละโมดูล
- ออกแบบส่วนติดต่อผู้ใช้ ออกแบบรูปแบบหน้าจอเพื่ออินเทอร์เฟส กับผู้ใช้โดยพยายามให้ใช้งานง่ายโดยคำนึงถึงความสะดวกและง่ายต่อการใช้งาน

5.3.2 การพัฒนาและการทดสอบส่วนย่อย

เขียนโปรแกรมที่ได้ทำการออกแบบไว้ และทำการทดสอบว่าโปรแกรมที่ได้สามารถทำงานได้หรือไม่ และถูกต้องตามที่กำหนดไว้หรือไม่

5.3.3 รวบรวมและทดสอบระบบ

นำโมดูลและส่วนประกอบต่าง ๆ ที่ได้ทำการเขียนและทดสอบว่าถูกต้องแล้วมารวมกัน เป็นโปรแกรมเดียว และทำการทดสอบโปรแกรมนั้นกับระบบจริง

5.3.4 ทำคู่มือประกอบการใช้งาน

บันทึกรายละเอียดและขั้นตอนการติดตั้งและใช้งานโปรแกรมที่สร้างขึ้นเพื่อเป็น เอกสารอ้างอิงประกอบกับการใช้งาน

5.3.5 ใช้งานและดูแล

เมื่อโปรแกรมทำงานได้อย่างถูกต้องแล้ว ก็จะถูกนำไปติดตั้งใช้งานจริง และในอนาคตหากมีการเปลี่ยนแปลงความต้องการบางอย่างก็จะต้องทำการแก้ไขพัฒนาโปรแกรมให้ได้ตามต้องการ

## 5.4 ปัญหาและอุปสรรคที่พบในขณะปฏิบัติงาน

1. ขาดแคลนเครื่องมือและอุปกรณ์ที่ใช้ในการทำโปรเจ็คเนื่องจากการพัฒนาในช่วงแรกนั้นทำได้ลำบาก เพราะจะต้องทำการทดลองกับเครือข่ายจริง ๆ ซึ่งแตกต่างจากการทดลองเขียนโปรแกรมทั่วไปที่สามารถเขียนได้โดยไม่ต้องติดต่อใช้งานกับระบบเครือข่าย
2. ขาดประสบการณ์ในการเขียนโปรแกรมขนาดใหญ่ ๆ ที่ถูกนำไปใช้งานจริง
3. มีปัญหาเรื่องการประมวลผลข้อมูลไม่ทันกับแพ็กเกจที่เข้ามาได้ทำให้บางครั้งค่าที่ได้ออกมาจากการคำนวณอาจจะคลาดเคลื่อนไปจากค่าที่เป็นจริงไปบ้าง ซึ่งในทางปฏิบัติจริง ๆ แล้วจะใช้ฮาร์ดแวร์เพื่อวิเคราะห์แทน

4. ขาดประสบการณ์ในการทำส่วนติดต่อแบบกราฟฟิก จึงทำให้เสียเวลาในการเขียนโปรแกรมมากกว่าการเขียนโปรแกรมอย่างอื่น

5. มีปัญหาเกี่ยวกับการเก็บข้อมูลของตัวแปรประเภทต่าง ๆ ทำให้เสียเวลาในการหาข้อผิดพลาด

# บทที่ 6
# บทวิจารณ์และบทสรุป

## 6.1 บทสรุปและวิจารณ์

โครงงานที่จัดทำขึ้นนี้เป็นโครงงานสำหรับช่วยในการศึกษาการทำงานภายในเครือข่ายเพื่อให้เข้าใจถึงการทำงานของโปรโตคอลในแต่ละเลเยอร์ และสามารถเฝ้าดูการใช้งานในระบบเครือข่ายเพื่อใช้ประกอบในการปรับปรุงและการพัฒนาระบบเครือข่าย โดยที่สามารถทราบถึงปริมาณการใช้งาน และข้อบกพร่องต่าง ๆ ที่เกิดขึ้นในระบบเครือข่าย เนื่องจากปัจจุบันการใช้งานคอมพิวเตอร์ (Computer) ในรูปแบบของระบบเครือข่าย (Network) ได้เข้ามามีบทบาทเพื่อช่วยเพิ่มประสิทธิภาพในการทำงานมากยิ่งขึ้น โดยเฉพาะในองค์กรขนาดใหญ่ ๆ จะสามารถใช้ประโยชน์จากการใช้ทรัพยากรร่วมกันหรือจากการทำงานเป็นระบบเครือข่าย จึงได้ทำการพัฒนาโปรแกรม (Program) ขึ้นเพื่อใช้เฝ้าดูและวิเคราะห์แพ็กเกจ (Packet) ในระบบเครือข่ายท้องถิ่น (LAN:Local Area Network) โดยทำการตรวจสอบแพ็กเกจที่ได้รับจากสื่อระบบที่ผ่านการทำงานของแพ็กเกจไดรเวอร์ (Packet Driver) ซึ่งจะรับแพ็กเกจจากการ์ดเชื่อมต่อเน็ตเวิร์ค (NIC:Network Interface Card) เข้ามาวิเคราะห์ถึงส่วนต่าง ๆ ที่จำเป็นต่อการศึกษาการทำงานและตรวจสอบระบบเครือข่าย โดยวิเคราะห์ถึง ปริมาณการใช้งาน (Utilization) การสนทนา (Conversation) โปรโตคอลดิสตริบิวชั่น (Protocol Distribution) เฟรมไซส์ดิสตริบิวชั่น (Framesize Distribution) และทำการวิเคราะห์โปรโตคอล (Protocol Analyser) แต่ในทางกลับกันอาจจะเป็นผลร้าย เนื่องจากอาจมีบุคคลที่ต้องการเข้ามาทำลายระบบ เนื่องมาจากข้อเสียเปรียบของระบบอินเทอร์เนตที่ข้อมูลจะไม่มีการเข้ารหัสไว้ ดังนั้นจึงทำให้บุคคลใดก็ตามที่สามารถเข้ามาดูข้อมูลที่ได้รับเข้ามาก็จะสามารถได้รับรหัสผ่านของผู้อื่นได้

โครงงานนี้ได้พัฒนามาจากโครงงานของปีที่แล้วโดยทำการเพิ่มส่วนต่าง ๆ จากเดิม และทำการแก้ไขการทำงานบางส่วนที่ไม่สมบูรณ์ ทำให้สามารถใช้งานได้ตามวัตถุประสงค์ที่ต้องการ ซึ่งมีรายละเอียดการพัฒนาโปรเจ็คดังนี้

- ส่วนติดต่อกับผู้ใช้
- วิเคราะห์โปรโตคอล
- กรองแพ็กเกจ
- Sampling แพ็กเกจ และทำการดูย้อนหลัง
- การกระจายของโปรโตคอล และขนาดของเฟรม
- การสนทนาระหว่างโฮส
- วิเคราะห์สถิติ
- การเฝ้าดูเฉพาะเครื่อง

## 6.2 แนวทางการพัฒนาต่อ

1. สามารถวิเคราะห์โปรโตคอลในแต่ละเลเยอร์ได้มากขึ้น โดยเฉพาะอย่างยิ่งโปรโตคอลที่ใช้ในเน็ตแวร์แลน ทำให้สามารถดูถึงการใช้งานต่าง ๆ ของเครือข่ายได้มากขึ้น เช่น ปริมาณการใช้งานของไฟล์เซิฟเวอร์ การใช้งานของพริ้นเตอร์ เพื่อนำไปใช้ในการทำบาลานซ์เซิฟเวอร์โหลด (balancing server load)

2. สามารถดักจับแพ็กเกจได้มีความถูกต้องเพิ่มขึ้น

3. สามารถทำการติดต่อได้อย่างมีประสิทธิภาพและมีความสะดวกเพิ่มขึ้น

4. สามารถพัฒนาไปใช้ในระบบวินโดว์ได้โดยผ่านทาง NDIS (Network Driver Interface Specification) ของวินโดว์

5. สามารถใช้งานกับระบบเครือข่ายชนิดอื่นได้เช่น โทเค็นริง เป็นต้น

6. สามารถทำการดึงข้อมูลจากเครือข่ายอื่น ๆ ได้โดยเรียกเอเจ้นท์ไว้

# ภาคผนวก ก.

## คำศัพท์ คำแปล และความหมาย

| คำศัพท์ | คำแปล | ความหมาย |
|---|---|---|
| access type | | ประเภทการใช้งาน |
| ACK | เอซีเค | บิตตอบรับใช้ในควบคุมทีซีพี |
| acknowledgement | | การตอบรับ |
| address | แอดเดรส | ที่อยู่ ตำแหน่ง |
| ARP | เออาร์พี | โปรโตคอลระดับเน็ตเวิร์คเลเยอร์ใช้ในการตรวจสอบว่าโฮสยังอยู่หรือไม่ |
| back-to-back receives | แบ็คทูแบ็ครีซีฟ | การรับในระดับล่างด้วยกัน |
| Bandwidth | แบนด์วิท | ความกว้างของช่องสัญญาณที่จะสื่อสารได้ |
| Banyan's | ไบยัน | เครือข่ายชนิดหนึ่ง |
| Basic Function | | ฟังก์ชันพื้นฐาน |
| Bit | บิต | หน่วยข้อมูลที่เล็กที่สุด |
| Bottlenecks | คอขวด | มีการใช้สื่อในการส่งข้อมูลมากไม่เพียงพอกับความต้องการของเครือข่าย |
| Bridges | บริดจ์ | อุปกรณ์ที่ใช้เชื่อมเน็ตเวิร์คสองวงเข้าด้วยกันทำงานในระดับดาต้าลิงค์เลเยอร์ |
| Broadcast | บรอดคาสท์ | การกระจายข้อมูลไปทั่ว |
| Broadcast medium | บรอดคาสท์มีเดียม | สื่อที่เป็นลักษณะที่ใช้ในการกระจายข้อมูล |
| buffer | บัฟเฟอร์ | ข้อมูลสำรอง |
| Checksum | เชคซัม | ค่าตรวจสอบ |
| class | คลาส | กลุ่มของชนิด |
| Client Server | | ระบบการให้บริการและรับบริการข้อมูล |
| Collecting | | ส่วนรวบรวมข้อมูล |
| Collision | คอลลิชัน | การชนกัน |

| | | |
|---|---|---|
| collision rate | | อัตราการชนกัน |
| communication service | คอมมิวนิเคชั่นเซอร์วิส | การบริการสื่อสาร |
| connector | คอนเน็คเตอร์ | ตัวเชื่อมต่อ |
| connectionless | คอนเน็คชั่นเลส | การเชื่อมต่อแบบไม่ตรวจสอบว่าส่งข้อมูลถึงหรือไม่ |
| control bit | คอนโทรลบิต | บิตควบคุม |
| corruption | คอรัปชั่น | เสียหาย |
| data offset | ดาต้าออฟเซต | ตำแหน่งข้อมูล |
| data throughput | ดาต้าทรูพุต | การไหลของข้อมูล |
| database | ดาตาเบส | ฐานข้อมูล |
| datagram | ดาต้าแกรม | หน่วยข้อมูลที่ใช้เรียกในเลเยอร์ดาต้าลิงค์ |
| Datalink layer | ดาต้าลิงค์เลเยอร์ | ระดับชั้นเชื่อมต่อข้อมูล เป็นเลเยอร์ที่ 2 ของโอเอสไอโมเดล |
| DECNET | เดคเน็ต | เครือข่ายชนิดหนึ่ง |
| demultiplexing | ดีมัลติเพล็กซิง | แยกส่วน |
| destination | เดสติเนชัน | ปลายทาง |
| destination port | | พอร์ตปลายทาง |
| dianostic | ไดอะนอสติค | ทดสอบ |
| DIX:DEC /Intel /Xerox | ดีไอเอ็กซ์ | องค์กรที่คิดค้นอีเทอร์เนต |
| DOS | ดอส | ระบบปฏิบัติการ |
| Driver parameter | ไดร์วเวอร์พารามิเตอร์ | ค่าที่ส่งแก่ไดร์วเวอร์ |
| electronics mail | อีเลคทรอนิคส์เมล์ | จดหมายในระบบเครือข่าย |
| End access | | สิ้นสุดการใช้งาน |
| end of list. | | สิ้นสุดรายชื่อ |
| End Systems | | ระบบปลายทาง |
| Error codes | | รหัสผิดพลาด |
| Error Rate | | อัตราการเกิดข้อผิดพลาด |
| Ethernet address | อีเทอร์เน็ตแอดเดรส | หมายเลขที่ระบุถึงสเตชันในเลเยอร์ดาต้าลิงค์ |
| Ethernet Frame | อีเทอร์เนตเฟรม | โปรโตคอลในระดับดาตาลิงค์ |
| Ethernet II | อีเทอร์เนต II | โปรโตคอลในระดับดาตาลิงค์ |
| Ethernet SNAP | อีเทอร์เนตสแนบ | โปรโตคอลในระดับดาตาลิงค์ |

| | | |
|---|---|---|
| ethertype value | | ค่าที่บ่งบอกถึงโปรโตคอลในเลเยอร์ถัดไปของอีเทอร์เนตเฟรม |
| event counters | | ตัวนับเหตุการณ์ |
| executable code | | โคดทำงาน |
| Extended Function | ฟังก์ชันเพิ่มเติม | เป็นฟังก์ชันเพิ่มเติมของแพ็กเกจไดรเวอร์ |
| External | เอกซ์เทอร์นอล | ภายนอก |
| failed packet rates | | อัตราแพ็กเก็จเสีย |
| FCS:Frame Check Sequence | เอฟซีเอส | ค่าซีอาร์ซีใช้ตรวจสอบแพ็กเกจ |
| FDDI | เอฟดีดีไอ | ระบบสื่อสารข้อมูลด้วยใยแก้วนำแสง |
| FIELD | ฟิลด์ | บริเวณ |
| file | ไฟล์ | เอกสาร |
| File Server | ไฟล์เซิฟเวอร์ | ผู้ให้บริการเอกสาร |
| Filter data | | ส่วนกรองข้อมูลโปรโตคอล |
| FIN | เอฟไอเอ็น | บิตควบคุมของ ทีซีพี |
| finger | ฟิงเกอร์ | โปรโตคอลใช้ตรวจสอบข้อมูล |
| firmware | เฟิร์มแวร์ | ข้อมูลที่เก็บอยู่ในรูปฮาร์ดแวร์ |
| flag | แฟลก | ค่าที่ใช้กำหนดควบคุมการทำงานต่างๆ |
| fragment offset | แฟรกเมนออฟเซต | ตำแหน่งที่แบ่งแยก |
| flow control | | ควบคุมการไหล |
| frame size distribution | | การกระจายของขนาดเฟรม |
| FTP | เอฟทีพี | โปรโตคอลที่ใช้รับส่งข้อมูล |
| FTP Software | เอฟทีพีซอฟต์แวร์ | องค์กรที่สร้างแพ็กเกจไดรเวอร์ |
| FTP_DATA | เอฟทีพีดาตา | โปรโตคอลที่ใช้รับส่งข้อมูล |
| gateway | เกตเวย์ | อุปกรณ์ใช้เชื่อมระหว่างองค์กร |
| Generating network traffic | | สร้างทราฟฟิคในเน็ตเวิร์ค |

| | | |
|---|---|---|
| Handler | แฮนเดลอร์ | ตัวถือ |
| header | เฮดเดอร์ | ส่วนหัวของแพ็กเกจ |
| High-performance Function | ฟังก์ชั่นประสิทธิภาพสูง | เป็นฟังก์ชั่นประสิทธิภาพสูงของแพ็กเกจไดรเวอร์ |
| Host Conversation | | สนทนาระหว่างโฮส |
| ICMP | ไอซีเอ็มพี | โปรโตคอลระดับทรานส์ปอร์ตเลเยอร์ใช้เพื่อแจ้งข้อมูลภายในเน็ตเวิร์ค |
| IEEE 802.2 | ไออีอีอี 802.2 | มาตรฐานโปรโตคอลดาต้าลิงค์ |
| IEEE 802.3 | ไออีอีอี 802.3 | มาตรฐานโปรโตคอลดาต้าลิงค์ |
| IEEE 802.5 | ไออีอีอี 802.5 | มาตรฐานโปรโตคอลดาต้าลิงค์ |
| incedence | อินซิเด้นซ์ | มาตรฐานโปรโตคอลดาต้าลิงค์ |
| Initiate access | | กำหนดเริ่มต้นใช้งาน |
| Integrated | อินทิเกรตเต็ด | การรวบรวม |
| Interface card | การ์ดอินเทอร์เฟส | อุปกรณ์เชื่อมต่อโครงข่าย |
| Interface Interrupt Vecter | อินเทอร์เฟสอินเทอร์รัพเวคเตอร์ | เป็นส่วนเชื่อมการใช้งานของแพ็กเกจไดรเวอร์โดยผ่านการทำงานของอินเทอร์รัพคอมพิวเตอร์ |
| interrupt | อินเทอร์รัพ | ขัดจังหวะ |
| IP | ไอพี | โปรโตคอลในเน็ตเวิร์คเลเยอร์ |
| IPX | ไอพีเอ็กซ์ | โปรโตคอลในเน็ตเวิร์คเลเยอร์ |
| Kilobyte per second | กิโลไบต์ต่อวินาที | อัตราปริมาณข้อมูลที่ใช้ |
| LAN:Local Area Network | แลน | เครือข่ายท้องถิ่น |
| layer | เลเยอร์ | ระดับชั้น |
| Lifenet's | ไลฟ์เน็ต | เครือข่ายชนิดหนึ่ง |
| load balancing | โหลดบาลานซิ่ง | การใช้งานให้สมดุล |
| local network headers | | ส่วนหัวของเน็ตเวิร์คเลเยอร์ |
| mainframe | เมนเฟรม | เครื่องคอมพิวเตอร์ขนาดใหญ่ |
| maximum segment size | | ขนาดสูงสุดของเซกเม้นท์ |
| maximum throughput | | การใช้งานสูงสุด |
| Megabit per second | เม็กกะบิตต่อวินาที | อัตราในการส่งข้อมูลในเน็ตเวิร์ค |
| minicomputer | มินิคอมพิวเตอร์ | เครื่องคอมพิวเตอร์ขนาดกลาง |

| | | |
|---|---|---|
| minimum delay | มินิมั่มดีเลย์ | ถ่วงเวลาน้อยที่สุด |
| minimum load times and levels | | เวลาและระดับการใช้งานต่ำสุด |
| module | โมดูล | ส่วนการทำงาน |
| monitoring | มอนิเตอร์ริง | เฝ้าดู |
| monitoring agent | มอนิเตอร์ริงเอเจนท์ | การทำงานเฝ้าดู |
| monitoring information | | ข้อมูลที่ใช้เฝ้าดู |
| MTU:Maximum Transmission Units | เอ็มทียู | ขนาดสูงสุดที่สามารถส่งแพ็กเกจในเน็ตเวิร์ค |
| multicast list | มัลติคาสลิสต์ | รายชื่อกลุ่มหมายเลขแอดเดรส |
| multiple connection | มัลติเปิลคอนเนคชัน | การเชื่อมต่อ |
| name server | เนมเซิฟเวอร์ | ผู้ให้บริการชื่อ |
| NECTEC | เน็คเทค | องค์กร |
| netware lan | เน็ตแวร์แลน | เครือข่ายท้องถิ่นที่เป็นเน็ตแวร์ |
| network | เน็ตเวิร์ค | เครือข่าย ประกอบด้วยสเตชันและสื่อที่ใช้ในการเชื่อมต่อ |
| network communication media | เน็ตเวิร์คคอมมิวนิเคชั่นมีเดีย | สื่อที่ใช้ในการเชื่อมต่อ |
| network component | | ส่วนประกอบเน็ตเวิร์ค |
| Network Conversation | เน็ตเวิร์คคอนเวอร์เซชั่น | การสนทนาภายในเน็ตเวิร์ค |
| network file system | เน็ตเวิร์คไฟล์ซิสเต็ม | ระบบเอกสารในเน็ตเวิร์ค |
| network interface card | เน็ตเวิร์คอินเทอร์เฟสการ์ด | อุปกรณ์ที่ใช้เชื่อมต่อกับเครือข่าย |
| network jamming | เน็ตเวิร์คแจมมิ่ง | ข้อมูลที่ส่งไปแล้วแต่มีการชนกัน |
| Network Layer | เน็ตเวิร์คเลเยอร์ | ระดับชั้นที่ 3 ของโอเอสไอโมเดล |
| network media | | สื่อเน็ตเวิร์ค |
| Network media's standard packet type | เน็ตเวิร์คมีเดียสแตนดาร์ดแพ็กเกจไทป์ | |
| network service | เน็ตเวิร์คเซอร์วิส | การบริการของเน็ตเวิร์ค |
| Network throughput | เน็ตเวิร์คทรูพุต | การใช้งานของเน็ตเวิร์ค |
| NIC:Network Interface Card | การ์ดเชื่อมต่อเน็ตเวิร์ค | อุปกรณ์ที่ใช้เชื่อมต่อกับเครือข่าย |
| NOP;No Operation | เอ็นโอพี | ไม่มีคำสั่ง |
| Novell's | โนเวล | องค์กร |
| Novell's IPX header | โนเวลไอพีเอ็กซ์เฮดเดอร์ | ส่วนหัวโปรโตคอลไอพีเอ็กซ์ |
| Novell's SPX header | โนเวลเอสพีเอ็กซ์เฮดเดอร์ | ส่วนหัวโปรโตคอลเอสพีเอ็กซ์ |

| number | นัมเบอร์ | หมายเลข |
|---|---|---|
| One's complement | วันคอมพลีเมนท์ | การกลับบิตข้อมูลจาก 0 เป็น 1 และ 1 เป็น 0 แล้วลบ 1 |
| optimum packet size | | ขนาดแพ็กเกจที่เหมาะสม |
| option | อ๊อปชั่น | ตัวเลือก |
| OSI | โอเอสไอ | ระบบเชื่อมต่อแบบเปิด |
| overhead | โอเวอร์เฮด | ส่วนสิ้นเปลือง |
| packet | แพ็กเกจ | ข้อมูลที่ใช้ส่งในสื่อเครือข่าย |
| Packet Driver | แพ็กเกจ ไดรเวอร์ | ส่วนติดต่อข้อมูลของการ์ดอินเตอร์เฟสกับโปรแกรม |
| Packet per second | | จำนวนแพ็กเกจต่อวินาที |
| Packet size distribution | | การกระจายของขนาดแพ็กเกจ |
| Peak | พีค | สูงสุด |
| peak load times and levels | | เวลาและระดับการใช้งานสูงสุด |
| physical address | ฟิซิคอลแอดเดรส | ตำแหน่งทางกายภาพ |
| platform | แพลตฟอร์ม | รูปแบบ |
| point-to-point communication link | พอย์นทูพอย์นคอมมิวนิเคชั่นลิงคื | การเชื่อมต่อระหว่างจุดหนึ่งไปยังจุดหนึ่ง |
| pointer | พอย์นเตอร์ | ตัวชี้ |
| port number | | หมายเลขพอร์ต |
| predecessor | พรีดีเซสเซอร์ | อันก่อน |
| protocol | โปรโตคอล | กฎเกณฑ์ในการเชื่อมต่อ |
| Protocol distribution | โปรโตคอลดิสตริบิวชัน | การกระจายการใช้งานของโปรโตคอล |
| protocol header | | ส่วนหัวของโปรโตคอล |
| protocol in use | | โปรโตคอลที่ใช้งานอยู่ |
| protocol number | โปรโตคอลนัมเบอร์ | หมายเลขโปรโตคอล |

| | | |
|---|---|---|
| Protocol Stack | โปรโตคอลสแต็ก | การใช้งานโปรโตคอลที่เรียกผ่านโปรโตคอลอื่นไปเป็นชั้น ๆ |
| Protocol use in conversation | | โปรโตคอลที่ใช้ในการสนทนา |
| PSH | พีเอสเฮส | บิตควบคุมในโปรโตคอลทีซีพี |
| Push Function | พุชฟังก์ชั่น | |
| query | คิวรี | |
| Raw data | รอ ดาต้า | ข้อมูลดิบรวมทั้งส่วนหัวและเอฟซีเอส |
| Real-time data analyzer | เรียลไทม์ดาต้าอานาไลเซอร์ | วิเคราะห์ข้อมูลแบบทันที |
| Real-time systems | เรียลไทม์ซิสเต็ม | ระบบตอบสนองทันที |
| Rearranging | ริอาร์เร้นท์จิ้ง | จัดเตรียมใหม่ |
| Receiver mode | | โหมดการรับ |
| Recording network errors | | เก็บข้อผิดพลาดในเน็ตเวิร์ค |
| Register | รีจิสเตอร์ | หน่วยความจำของคอมพิวเตอร์ในการทำงาน |
| Remote diagnostic capability | | วิเคราะห์ในระยะไกล |
| Remote execution | | การทำงานปลายทาง |
| Remote login | รีโมทล็อกอิน | |
| Remote printing | | การพิมพ์ปลายทาง |
| Remote session | รีโมทเซสชั่น | การประชุมทางไกล |
| Request | รีเควส | การร้องขอ |
| Response | เรสปอน | ตอบสนอง |
| Response time of the file server processes | | เวลาตอบสนองของโปรเซสไฟล์เซิฟเวอร์ |
| RLOGIN | อาร์ล็อกอิน | โปรโตคอลในชุดทีซีพี/ไอพี |
| Router | เราท์เตอร์ | อุปกรณ์ใช้เชื่อมต่อเครือข่าย |
| RPC: Remote Procedure Call | รีโมทโพรซีเดอร์คอล | โปรโตคอลชนิดหนึ่ง |
| RST | อาร์เอสที | บิตควบคุมของทีซีพี |
| Runts | รั้น | แพ็กเกจขนาดสั้นกว่าปกติ |
| Sensing | เซนซิ่ง | ส่วนรับข้อมูล |
| Sequence | ซีเคว้น | หมายเลขลำดับ |
| Serial line MODEM | ซีเรียลไลน์โมเด็ม | |
| Server process | เซิฟเวอร์โปรเซส | การทำงานของระบบบริการ |

| | | |
|---|---|---|
| Service access point field | เซอร์วิสแอคเซสพอยน์ฟิลด์ | |
| Shared states | แชร์เสตท | สถานะร่วม |
| Sliding window | สไลด์ดิ้งวินโดว์ | การใช้งานของทีซีพีโดยแบ่งการตอบรับข้อมูลเป็นช่วง ๆ |
| Source | ซอส | ต้นทาง |
| Source port | | พอร์ตต้นทาง |
| Specification | สเปคซิฟิเคชั่น | ข้อกำหนด |
| SPX | เอสพีเอ็กซ์ | โปรโตคอลระดับทรานส์สปอร์ต |
| Stack Protocol | สแต็คโปรโตคอล | การใช้งานของโปรโตคอลที่เรียกกันเป็นชั้น ๆ |
| State | สเตต | สถานะ |
| Storage media | | สื่อบรรจุ |
| SYN | เอสวายเอ็น | บิตควบคุมของทีซีพี |
| Table offset | เทเบิ้ลออฟเซต | ตำแหน่งตาราง |
| TCP | ทีซีพี | โปรโตคอลระดับทรานส์สปอร์ต |
| TCP/IP | ทีซีพี/ไอพี | กลุ่มของโปรโตคอลที่ใช้ทีซีพีและไอพี |
| TELNET | เทลเนต | โปรโตคอลในระดับทรานส์สปอร์ตเลเยอร์ |
| Terminal | เทอร์มินอล | คอมพิวเตอร์ปลายทาง |
| Terminal server | เทอร์มินอลเซิฟเวอร์ | บริการคอมพิวเตอร์ปลายทาง |
| Time to live | | เวลาที่ยังอยู่ |
| Total bandwidth | | แบนด์วิททั้งหมด |
| Traffic | ทราฟฟิค | การจราจร |
| Transmit | ทรานสมิต | การส่ง |
| Transmit strategies | | กลยุทธ์การส่ง |
| Transport layer | ทรานสปอร์ตเลเยอร์ | ระดับชั้นที่ 4 ในโอเอสไอโมเดล |
| Type | ไทป์ | ชนิด |
| UDP | ยูดีพี | โปรโตคอลในระดับทรานส์สปอร์ตเลเยอร์ |
| UDP protocol number | | หมายเลขยูดีพีโปรโตคอล |
| Unresponsive nodes | | โหนดที่ไม่ตอบสนอง |
| Upper layer | อัพเปอร์เลเยอร์ | ระดับชั้นบน |

| | | |
|---|---|---|
| URG | ยูอาร์จี | บิตควบคุมของทีซีพี |
| Urgent field | ฟิลด์เออเจ้นท์ | ช่วง |
| Urgent pointer | เออเจ้นท์พอยน์เตอร์ | ตำแหน่งของเออเจ้นท์ |
| Usable data | | ข้อมูลที่ใช้จริง |
| Usable data throughput | ยูซเซเบิ้ลดาต้าทรูพุต | การใช้งานข้อมูลที่ใช้งานจริง |
| Utilize | ยูทิไลซ์ | อัตราการใช้งาน |
| Workgroups | เวิร์คกรุป | การทำงานโดยรวมกลุ่ม |
| Workstation | เวิร์คสเตชั่น | เครื่องที่ใช้งาน |
| XNS | เอ็กซ์เอ็นเอส | โปรโตคอลชนิดหนึ่ง |

# ภาคผนวก ข

## รูปแบบส่วนหัวของแต่สะโปรโตคอล

| Ethernet II Frame | |
|---|---|
| Destination Address | 6 ไบต์ |
| Source Address | 6 ไบต์ |
| Type | 2 ไบต์ |
| Data | 46-1500 ไบต์ |

| Ethernet 802.3 Frame | |
|---|---|
| Destination Address | 6 ไบต์ |
| Source Address | 6 ไบต์ |
| Length | 2 ไบต์ |
| Data | 46-1500 ไบต์ (เริ่มต้นด้วย 0xFFFF) |

| Ethernet 802.2 Frame | |
|---|---|
| Destination Address | 6 ไบต์ |
| Source Address | 6 ไบต์ |
| Type | 2 ไบต์ |
| DSAP | 1 ไบต์ |
| SSAP | 1 ไบต์ |
| Control | 1 ไบต์ |
| Data | 43-1497 ไบต์ |

| Ethernet SNAP Frame | |
|---|---|
| Destination Address | 6 ไบต์ |
| Source Address | 6 ไบต์ |
| Type | 2 ไบต์ |
| DSAP | 1 ไบต์ |
| SSAP | 1 ไบต์ |

| | |
|---|---|
| Control | 1 ไบต์ |
| Organization Code | 3 ไบต์ |
| Ethernet Type | 2 ไบต์ |
| Data | 38-1492 ไบต์ |

| IPX Packet | |
|---|---|
| Check Sum | 2 ไบต์ |
| Length | 2 ไบต์ |
| Transport | 1 ไบต์ |
| Packet Type | 1 ไบต์ |
| Destination Network | 4 ไบต์ |
| Destination Host | 6 ไบต์ |
| Destination Socket | 2 ไบต์ |
| Source Network | 4 ไบต์ |
| Source Host | 6 ไบต์ |
| Source Socket | 2 ไบต์ |
| Data | |

| IP Packet | |
|---|---|
| Version + Header Length | 1 ไบต์ (1/2 ไบต์ + ? ไบต์) |
| Type Of Service | 1 ไบต์ |
| Length | 2 ไบต์ |
| Identifier | 2 ไบต์ |
| Flag | 3 บิต |
| Fragment Offset | 13 บิต |
| Time to live | 1 ไบต์ |
| Protocol | 1 ไบต์ |
| Header Check Sum | 2 ไบต์ |
| Source Address | 4 ไบต์ |
| Destination Address | 4 ไบต์ |
| Option | Variable |
| Data | |

| ARP Packet | |
|---|---|
| Hardware Type | 2 ไบต์ |
| Protocol Type | 2 ไบต์ |
| Hardware Address Length | 1 ไบต์ |
| Protocol Address Length | 1 ไบต์ |
| Operation Code | 2 ไบต์ |
| Send Hardware Address | 6 ไบต์ |
| Send Protocol Address | 4 ไบต์ |
| Target Hardware Address | 6 ไบต์ |
| Target Protocol Address | 4 ไบต์ |

| SPX Packet | |
|---|---|
| Control | 1 ไบต์ |
| Data Type | 1 ไบต์ |
| Source ID | 2 ไบต์ |
| Destination ID | 2 ไบต์ |
| Sequence Number | 2 ไบต์ |
| Acknowledgement Number | 2 ไบต์ |
| Allocation Number | 2 ไบต์ |
| Data | |

| NCP Request Packet | |
|---|---|
| Request Type | 2 ไบต์ |
| Sequence Number | 1 ไบต์ |
| Connection Number Low | 1 ไบต์ |
| Task Number | 1 ไบต์ |
| Connection Number High | 1 ไบต์ |
| Data | |

| NCP Reply Packet | |
|---|---|
| Request Type | 2 ไบต์ |
| Sequence Number | 1 ไบต์ |

| | |
|---|---|
| Connection Number Low | 1 ไบต์ |
| Task Number | 1 ไบต์ |
| Connection Number High | 1 ไบต์ |
| Completion Code | 1 ไบต์ |
| Connection Status | 1 ไบต์ |
| Data | |

| ICMP Packet | |
|---|---|
| Type | 1 ไบต์ |
| Code | 1 ไบต์ |
| Check Sum | 2 ไบต์ |
| Data | |

**ICMP Message Types**

| ชนิด | ความหมาย |
|---|---|
| 0 | Echo Reply |
| 3 | Destination Unreachable |
| 4 | Source Quench |
| 5 | Redirect |
| 8 | Echo |
| 11 | Time Exceeded |
| 12 | Parameter Problem |
| 13 | Timestamp |
| 14 | Timestamp Reply |
| 15 | Information Request |
| 16 | Information Reply |

| Echo and Echo Reply Message | |
|---|---|
| Type | 1 ไบต์ (0 or 8) |
| Code | 1 ไบต์ |
| Check Sum | 2 ไบต์ |
| Identifier | 2 ไบต์ |
| Sequence Number | 2 ไบต์ |

| Data | |
|---|---|

| Information and Information Reply Message | |
|---|---|
| Type | 1 ไบต์ (15 or 16) |
| Code | 1 ไบต์ |
| Check Sum | 2 ไบต์ |
| Identifier | 2 ไบต์ |
| Sequence Number | 2 ไบต์ |

| Destination Unreachable Message + Source Quench Message + Time Exceeded Message | |
|---|---|
| Type | 1 ไบต์ (3,4 หรือ 11) |
| Code | 1 ไบต์ |
| Check Sum | 2 ไบต์ |
| Unused | 4 ไบต์ |
| Data | |

| Redirect Message | |
|---|---|
| Type | 1 ไบต์ (5) |
| Code | 1 ไบต์ |
| Check Sum | 2 ไบต์ |
| Gateway Internet Address | 4 ไบต์ |
| Data | |

| Parameter Problem Message | |
|---|---|
| Type | 1 ไบต์ (12) |
| Code | 1 ไบต์ |
| Check Sum | 2 ไบต์ |
| Pointer | 1 ไบต์ |
| Unused | 3 ไบต์ |
| Data | |

| Timestamp and Timestamp Reply Message | |
|---|---|
| Type | 1 ไบต์ (13 หรือ 14) |

| | |
|---|---|
| Code | 1 ไบต์ |
| Check Sum | 2 ไบต์ |
| Original Timestamp | 4 ไบต์ |
| Receive Timestamp | 4 ไบต์ |
| Transmit Timestamp | 4 ไบต์ |

| TCP Packet | |
|---|---|
| Source Port | 2 ไบต์ |
| Destination Port | 2 ไบต์ |
| Sequence Number | 4 ไบต์ |
| Acknowledgement Number | 4 ไบต์ |
| Header Length | 1 ไบต์ |
| Code Bits | 1 ไบต์ |
| Windows | 2 ไบต์ |
| Check Sum | 2 ไบต์ |
| Urgent Pointer | 2 ไบต์ |
| Option | Variable |
| Data | |

| UDP Packet | |
|---|---|
| Source Port | 2 ไบต์ |
| Destination Port | 2 ไบต์ |
| Length | 2 ไบต์ |
| Check Sum | 2 ไบต์ |
| Data | |

| RIP Packet | |
|---|---|
| Command | 1 ไบต์ |
| Version | 1 ไบต์ |
| Zero | 2 ไบต์ |
| Address Family ID | 2 ไบต์ |
| Zero | 2 ไบต์ |
| IP Address | 4 ไบต์ |

| | |
|---|---|
| Zero | 8 ไบต์ |
| Distance to network n | 4 ไบต์ |
| Can Repeat from first Address Family ID fields | |

# ภาคผนวก ค
# การกำหนดตัวเลขในอีเทอร์เนต

ในการใช้งานของอีเทอร์เนตจะต้องมีการกำหนดค่าต่างๆ เหล่านี้เพื่อให้เป็นมาตรฐานและสามารถใช้งานร่วมกันได้ ซึ่งต้องมีส่วนต่างๆ ดังนี้ ไทป์โคด , เวนเดอร์โคด (vendorcodes) , มัลติคาสท์ (รวมถึงบอร์ดคาสท์) แอดเดรส อีเทอร์เนตไทป์ ตำแหน่งอ๊อกเตด ที่ 13 และ 14 ของอิเทอร์เนตเป็นส่วนของ อีเทอร์ไทป์ หรือ IEEE 802.3 เส้นท์ ซึ่งอีเทอร์ไทป์ นั้น ซีรอก (XEROX) เป็นผู้รับผิดชอบอยู่ บางค่าอาจเป็นมาตรฐานจะใส่เครื่องหมาย " + " นอกนั้นถูกกำหนดเพื่อใช้เป็นการส่วนตัว ซึ่งข้อมูลนี้จะมีส่วนที่มาจาก ซีรอกพับลิกเทอร์เนตแพ็กเกตไทป์ (Xerox Public Ethernet Packet Type) , IEEE 802.3 สแตนด์ดาร์ด , และจากเน็ตเวิร์ค เมเนจเจอร์ และ เวนเดอร์

**โปรโตคอลอีเทอร์เนตไทป์ (Protocol Ethernet Type)**

| | |
|---|---|
| @ 0000-05DC | IEEE802.3 Length Field (0...1500.) |
| + 0101-01FF | Experimental |
| 0200 | Xerox PUP (conflicts with 802.3 Length Field range) |
| 0201 | Xerox PUP Address Translation (conflicts ....) |
| 0400 | Xixdorf (conflict with 802.3 Length Field) |
| +*0800 | DOD Internet Protocol (IP) |
| + 0805 | X 25 Level 3 |
| +*0806 | Address Resolution Protocol (ARP) (for IP) |
| 8005 | HP Probe protocol |
| + 8019 | Apollo DOMAIN |
| + 8035 | Reverse Address Resolution Protocol (RARP) |
| 8037 | IPX (Novell Netware) |
| + 809B | Ether Talk (Apple Talk over Ethernet ) |
| 80D5 | IBM SNA Services over Ethernet |
| + 80F3 | Apple Talk Address Resolution Protocol (AARP) |
| + 8173 | Novell (old) NetWare IPX (ECONFIG E option) |
| + 8138 | Novell , Inc. |
| 814C | SNMP over Ethernet (see RFC 1089) |
| 817D | XTP |
| 83DD | IP version 6 |
| 8888 | HP LanProbe test? |

| | | |
|---|---|---|
| + 9000 | Loopback (Configuration Test Protocol) | |
| AAAA | DECNET? | Used by VAX 6220 DEBNI |
| % FFF0 | BBN VITAL-LanBridge cache wakeups | |

<u>หมายเหตุ</u>

- ตัวเลขเป็นฐาน 16
- "*" = โปรโตคอล นี้ถูกใช้ในการทำอีเทอร์เน็ตบอร์ดคาสท์ ซึ่งอาจจะใช้ในมัลติคาสท์ ก็ได้
- " % " = อาจจะนำไปใช้ส่วนตัว คือ ยังไม่ได้ลงทะเบียน
- " + " = โปรโตคอล ซึ่งถูกอ้างอิงโดยซีรอกในหนังสือ " COURIER (page 8-9 ) October 1988 issue of ในหัวข้อ publicly assigned numbers
- "@" = ตามที่อธิบายใน COURIER (page 8) "ถ้ามีขนาดน้อยกว่า 600 H packet จะเป็นของ 802.3 และถ้าใหญ่กว่า 600 H จะถือว่าเป็น flag และ Ethernet packet

**เวนเดอร์โค้ด**

อีเทอร์เน็ตฮาร์ดแวร์แอดเดรส หรือ แมคประกอบด้วย 48 บิต หรือ 12 เฮกสาเดซิมอลดิจิต (Hexadecimal digits) (ตัวเลข 0-9 รวม A-F) ประกอบด้วย ช่วงแรกทางซ้าย 6 ดิจิต ซึ่งจะตรงกับหมายเลขของเวนเดอร์และที่เหลืออีก 6 ดิจิต เป็นซีเรียลนัมเบอร์ (Serial number ) ของอินเตอร์เฟสนั้นจากเวนเดอร์

อีเทอร์เนตแอดเดรสมักจะเขียนในรูปของการแบ่ง 2 เฮกดิจิต เพื่อในการอ่านและอ้างอิงได้ง่าย เช่น 12-34-56-78-9A-BC

เลขเหล่านี้เป็นตัวกำหนดถึงฟิซิคอลสเตชั่นแอดเดรส ที่ไม่ใช้มัลติคาสท์หรือบอร์ดคาสท์ ดังนั้น ดิจิตที่สองจากทางซ้าย จะเป็นเลขคู่ ไม่ใช้เลขคี่

Cisco = 00000C

Fujitsu = 00000E

NeXT = 00000F

Novell = 00001B

ATT = 00003D

Nokia = 00004B

NEC = 00004C

ATT = 000055

MIPS = 00006B

Sanyo = 0000A0 /* Sanyo Electronics

Xerox = 0000AA

Apollo = 000AC /* Apollo

HP ON = 0000C6 /* HP Intgnt Networks Oper (EON) * /

DEC = 0000F8 /* Digital Equipment Corporation */

IEE802 = 000143 /* IEEE802

3com = 0020AF

3com = 00608C

3Com = 00608C

CNET = 0080AD

NET = 0080B2

IEEE = 0080C2 /* IEEE 802.1 Committee */

Intel = 00AA00

3Com = 026060

3Com = 02608C

Bridge = 080002

Apple = 080007

HP = 080009

Apollo = 08001E

Sharp = 08001F /* Sharp */

Sun = 080020

NBI = 080022

FujiXe = 080037 /* Fuji Xerox */

Mortrla = 08003E /* Motorola */

Sony = 080046

IBM = 08005A /* (bit-reverse from Token-Ring) */

ATT = 08006A

Mitsu = 080070 /* Mitsubishi */

Casio = 080074 /* Casio */

SilicG = 080079 /* Silicon Graphics */

Xyplex = 080087

ATT = 09006A /* AT&T Use in smart hub */

IBM = 10005A /* not bit-reverse from Token-Ring) */

DECnet = 1000D4 /* DEC */

ApplUX = 1000E0 /* Apple A/UX (modified address for licenlising ) */

ATT = 800010 /* AT&T */

DECnet = AA0000

DECnet = AA0003

**บอร์ดคาสท์แอดเดรสและสเตชั่นแอดเดรส (Broadcast Address & station Address)**

| address | type | owner |
|---|---|---|
| FF-FF-FF-FF-FF-FF | 0600 | XNS packets, Hello or gateway search? 6 packets every 15 seconds, per XNS station |
| FF-FF-FF-FF-FF-FF | 0800 | IP (e.g. RWHOD via UDP) as needed |
| FF-FF-FF-FF-FF-FF | 0806 | ARP (for IP and CHAOS) as needed |
| FF-FF-FF-FF-FF-FF | 1600 | VALID packets , Hello or gateway search? 1 packets every 30 seconds, per VALID station |
| FF-FF-FF-FF-FF-FF | 8035 | Reverse ARP |
| FF-FF-FF-FF-FF-FF | 809b | EtherTalk |

| | |
|---|---|
| station 09001E000000 | " Apollo_DOMAIN " |
| station 090007FFFFFF | " Atalk_Broadcast " |
| station 0180C2000000 | " Bridge_Group_Addr " |
| station FFFFFFFFFFFF | " Broadcast " |
| station 09002B230000 | " DEC_Argo_Console " |
| station 09002B010000 | " DEC_Bridges " |
| station AB0000020000 | " DEC_Console " |
| station 09002B000006 | " DEC_Encryption " |
| station AB0000040000 | " DEC_END_nodes " |
| station 09002B040000 | " DEC_LAST " |
| station AB0003000000 | " DEC_LAT " |
| station 09002B00000F | " DEC_LAT_Units " |
| station 09002B010001 | " DEC_lv1_Bridges " |
| station AB0000030000 | " DEC_lv1_Router " |
| station 09002B020000 | " DEC_lv2_Router " |
| station 09002B000000 | " DEC_Mumps " |
| station 09002B020100 | " DEC_Name_Advert " |
| station 09002B020101 | " DEC_Name_Solicit " |
| station 09002B000007 | " DEC_Netbios " |
| station AB0000010000 | " DEC_Pmp/Load " |
| station 09002B000003 | " DEC_Traffic_Mon " |
| station 09002B000004 | " DEC_VAXELN " |
| station C00000000100 | " Ethernet Broadcast " |

| | |
|---|---|
| staion 0900090000004 | " HP_DLC " |
| station 090009000001 | " HP_Probe " |
| station 09002B000004 | " ISO_END_Stns " |
| station 09002B000005 | " ISO_Int_Stns " |
| station CF0000000000 | " Loopback " |
| station 090014000101 | " NCP_30_Servers " |
| station 030000000001 | " NetBIOS " |
| station 090002040002 | " Vtlink_Bridges " |
| station 09007C020005 | " Vtlink_Diag " |
| station 09007C010001 | " Vtlink_DLS " |
| station 09007C010004 | " Vtlink_DLS/NonDLS " |
| station 09007C010002 | " Vlink_DLS_Hello " |
| station 09007C010003 | " Vlink_DLS_Inlink " |
| station 090002040001 | " Vlink_Printers " |
| station 09007C050002 | " Vlink_Validation " |
| station C00000000001 | " Active Mon. " |
| station C00000000100 | " All Bridges " |
| station FFFFFFFFFFFF | " All Fs Broadcast " |
| station 800143000000 | " Bridge Group " |
| staion C000FFFFFFFF | " Broadcast " |
| station C00000000010 | " Config Srv " |
| station C00000000008 | " Error Mon. " |
| station C00000002000 | " LAN Manager " |
| station C00000000080 | " NetBIOS " |
| station C00000800000 | " NetWare " |
| station C00000000002 | " Param Server " |

**กำหนดอินเตอร์เนตโปรโตคอลนัมเบอร์**

| Decimal | Keyword | Protocol. | References |
|---|---|---|---|
| 0 | Reversed | | [JBP] |
| 1 | ICMP | Internet Control Message | [RFC792,JBP] |
| 2 | IGMP | Internet Group Management | [RFC1112,JBP] |
| 3 | GGP | Gateway-to-Gateway | [RFC823,MB] |

| | | | |
|---|---|---|---|
| 4 | IP | IP in IP (eneasolution) | [JBP] |
| 6 | TCP | transmission Control | {RFC793,JBP] |
| 11 | NVP-11 | Network Voice Protocol | [RFC741,SC3] |
| 17 | UDP | User Datagram | [RFC768,JBP] |
| 29 | ISO-TP4 | ISO Transport Protocol Class 4 | [RFC905,RC77] |
| 41 | SIP | Simple Internet Protocol | [SXD] |
| 55-60 | | Unassigned | [JBP] |
| 61 | | any host internal protocol | [JBP] |
| 62 | CFTP | CFTP | [CFTP,HCF2] |
| 63 | | any local network | [JBP] |
| 68 | | any distributed file system | [JBP] |
| 89 | OSPFIGP | OSPFIGP | [RFC1583, JTM4] |
| 91 | LARP | Locus Address Resolution protocol | [BXH] |
| 92 | MTP | Multicast Tranlsport Protocol | [SXA] |
| 94 | IPIP | Ip-within-IP Encapsulation Protocol | [JI6] |
| 97 | ETHERIP | Ether-within-IP Encapsulation | [RXHI] |
| 98 | ENCAP | Encapsulation Header | [RFC1241,RXB3] |
| 99 | | any private encryption scheme | [JBP] |
| 101-254 | | Unassigned | [JBP] |
| 255 | | Reserved | [JBP] |

**คลาสและไทป์ต่าง ๆในแพ็กเกจไดรเวอร์**

| | |
|---|---|
| DEC/Intel/Xerox "Bluebook" Ethernet | Class 1 |
| 3COM 3C500/3C501 | 1 |
| 3COM 3C505 | 2 |
| Interlan Ni5010 | 3 |
| BICC Data Networks 4110 | 4 |
| BICC Data Networks 4117 | 5 |
| MICOM-Interlan NP600 | 6 |
| Ungermann-Bass PC-NIC | 8 |
| Univation NC-516 | 9 |
| TRW PC-2000 | 10 |
| Interlan Ni5210 | 11 |
| 3COM 3C503 | 12 |

| | |
|---|---|
| 3COM 3C523 | 13 |
| Western Digital WD 8003 | 14 |
| Spider System S4 | 15 |
| Torus Frame Level | 16 |
| 10NET Communications | 17 |
| Gateway PC-bus | 18 |
| Gateway AT-bus | 19 |
| Gateway MCA-bus | 20 |
| IMC Pcnic | 21 |
| IMC Pcnic II | 22 |
| IMC Pcnic 8bit | 23 |
| Tigan Communications | 24 |
| Micromatic Research | 25 |
| Clarkson " Multiplexor" | 26 |
| D-Link 8-bit | 27 |
| D-Link 16-bit | 28 |
| D-Link PS/2 | 29 |
| Research Machine 8 | 30 |
| Research Machine 16 | 31 |
| Research Machine MCA | 32 |
| Radix Microsys. EXMI 16-bit | 33 |
| Interlan Ni9210 | 34 |
| Interlan Ni6510 | 35 |
| Vestra LANMASTER 16-bit | 36 |
| Vestra LANMASTER 8-bit | 37 |
| Allied Telesis PC/XT/AT | 38 |
| Allied Telesis NEC PC-98 | 39 |
| Allied Telesis Fujitsu FMR | 40 |
| Ungermann-Bass NIC/PS2 | 41 |
| Tiara LANCard/E AT | 42 |
| Tiara LANCard/E MC | 43 |
| Tiara LANCard/E TP | 44 |
| Spider Comm. SpiderComm8 | 45 |
| Spider Comm. SpiderComm16 | 46 |

| | |
|---|---|
| AT&T Starlan NAU | 47 |
| AT&T Starlan-10 NAU | 48 |
| AT&T Ethernet NAU | 49 |
| Intel smart card | 50 |
| ProNET-10 | Class2 |
| Proteon p1300 | 1 |
| Proteon p1800 | 2 |
| IEEE 802.5/ProNET-4 | Class 3 |
| IBM Token ring adapter | 1 |
| Proteon p1340 | 2 |
| Proteon p1344 | 3 |
| Gateway PC-bus | 4 |
| Gateway AT-bus | 5 |
| Gateway MCA-bus | 6 |
| Omninet | Class 4 |
| Appletalk | Class 5 |
| Serial line | Class 6 |
| Clarkson 8250-SLIP | 1 |
| Clarkson "Multiplexor" | 2 |
| Starlan | Class 7 |
| AreNet | Class 8 |
| Datapoint RIM | 1 |
| AX.25 | Class 9 |
| KISS | Class 10 |
| IEEE 802.3 w/802.2 hdrs | Class 11 |

| | |
|---|---|
| FDDI w/802.23 hdrs | Class 12 |
| Internet X.25 | Class 13 |
| Western Digital | 1 |
| Fronrtier Technology | 2 |
| N.T. LANSTAR(eneapsulating DIX) | Class 14 |
| NT LANSTAR/8 | 1 |
| NT LANSTAR/MC | 2 |

# เอกสารอ้างอิง

## เอกสารอ้างอิงที่เป็นหนังสือภาษาอังกฤษ

FTP Software, " PC/TCP packet driver specification & programming " ,IEEE Trans. On Broadcast, Vol 1.09 , 66p., 1994

Laura A. Chappel and Dan E. Hakes, "Netware LAN Analysis", SYBEX Inc ., 693 p., 1994

Novell, "LANalyzer for Windows 2.2 Installation and User's Guide", Novell Inc., 244 p., 1996

Theakson, " Netware LANs Performance and Troubleshooting " , ADDISON-WESLEY , 381 p., 1995

Dah Ming Chiu and Ram Sudama, "NETWORK MONITORING EXPLAINED Design and Application", ELLIS HORWOOD, 207 p., 1992

Ed Taylor , "Internetworking Handbook", McGraw-HILL, Inc, pp. 95-450.

## เอกสารอ้างอิงที่เป็นวารสารภาษาอังกฤษ

Jon Postel , "Transmission Control Protocol", RFC 793 , USC/Information Sciences Institute , 1981 , pp. 1-19.

Jon Postel , "User Datagram Control Protocol" , RFC 768 , USC/Information Sciences Institute , 1980 , pp. 1-3.

Jon Postel , "Internet Protocol" , RFC 760 , USC/Information Sciences Institute , 1980.

Jon Postel , "Assigned Number" , RFC 762 , USC/Information Sciences Institute , 1980.

Gabriel Ciobotaru , "GRAFICA Ver. 2.0" , Grafica™ , 1996.

กิตติกรรมประกาศ

ต้องขอขอบคุณพ่อและแม่ที่ให้กำเนิดเรามารวมถึงให้ความสนับสนุนทั้งทางด้านการเรียน การเงิน รวมถึงเป็นกำลังใจอันสำคัญที่ช่วยให้การทำโครงงานในครั้งนี้สำเร็จลุล่วงไปด้วยดี

ขอขอบคุณอาจารย์ที่ปรึกษา (อาจารย์ธนา หงษ์สุวรรณ) ที่คอยให้คำปรึกษา และให้คำแนะนำต่าง ๆ เกี่ยวกับโครงงานนี้ เป็นอย่างมาก ทำให้โครงงานในครั้งนี้สามารถลุล่วงไปด้วยดี รวมถึงอาจารย์ผู้คอยประสิทธิประสาทวิชาให้แก่พวกเราทุกท่าน และบุคคลที่ไม่ได้กล่าวถึงอีกมาก

ขอขอบคุณรุ่นพี่ที่ทำโครงงานนี้เพื่อเป็นแนวทางต่าง ๆ ให้สามารถทำงานได้ง่ายขึ้น

ขอบใจแด่เพื่อน ๆ ทุกคนที่คอยเป็นกำลังใจให้กับพวกเราเสมอมารวมถึงขอขอบใจน้อง ๆ ทุกคนที่คอยเป็นกำลังใจให้รวมถึงช่วยในการทำปริญญานิพนธ์ฉบับนี้ทำให้สามารถทำได้สำเร็จตามที่ต้องการ

ขอขอบคุณทุกท่านที่มีส่วนร่วมให้โครงงานนี้สำเร็จด้วยดี รวมถึงขอบคุณตัวเองที่มีความอดทนทำโครงงานนี้จนสำเร็จได้